(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2004-523035
(P2004-523035A)
(43)公表日　平成16年7月29日(2004.7.29)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 13/00	G06F 13/00	353A	5B089
H04L 12/56	H04L 12/56	200Z	5K030

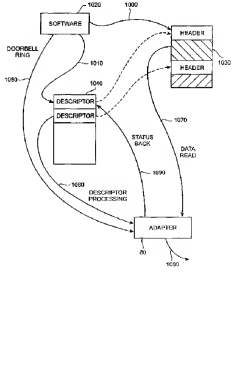
審査請求 有　予備審査請求 有　（全 74 頁）

(21)出願番号	特願2002-561695(P2002-561695)	(71)出願人	390009531
(86)(22)出願日	平成13年1月31日(2001.1.31)		インターナショナル・ビジネス・マシーンズ・コーポレーション
(85)翻訳文提出日	平成15年7月28日(2003.7.28)		INTERNATIONAL BUSINESS MASCHINES CORPORATION
(86)国際出願番号	PCT/IB2001/000122		
(87)国際公開番号	WO2002/061592		
(87)国際公開日	平成14年8月8日(2002.8.8)		アメリカ合衆国１０５０４　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
(81)指定国	AP(GH,GM,KE,LS,MW,MZ,SD,SL,SZ,TZ,UG,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,CH,CY,DE,DK,ES,FI,FR,GB,GR,IE,IT,LU,MC,NL,PT,SE,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BY,BZ,CA,CH,CN,CR,CU,CZ,DE,DK,DM,DZ,EE,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NO,NZ,PL,PT,RO,RU,SD,SE,SG,SI,SK,SL,TJ,TM,TR,TT,TZ,UA,UG,US,UZ,VN,YU,ZA,ZW	(74)代理人	100086243
			弁理士　坂口　博
		(74)代理人	100091568
			弁理士　市位　嘉宏
		(74)代理人	100108501
			弁理士　上野　剛史

最終頁に続く

(54)【発明の名称】メモリを介してデータ処理システムの間でデータのフローを制御する方法および装置

(57)【要約】
【課題】第１および第２のデータ処理システムの間でのメモリを介するデータのフローを制御する装置を提供すること。
【解決手段】この装置には、メモリ内の位置と第２データ処理システムの間で通信されるデータ・パケットを定義するフレーム記述子およびメモリ内の位置を識別するポインタ記述子を含む複数の記述子を生成する記述子ロジックが含まれる。この装置には、第１および第２のデータ処理システムによるアクセスのために、記述子ロジックによって生成された記述子を保管する記述子テーブルも含まれる。



【特許請求の範囲】
【請求項１】
メモリを介して第１および第２のデータ処理システムの間でデータのフローを制御する装置であって、メモリ内の位置と第２データ処理システムとの間で通信されるデータ・パケットを定義するフレーム記述子およびメモリ内の位置を識別するポインタ記述子を含む複数の記述子を生成する記述子ロジックと、第１および第２のデータ処理システムによるアクセスのために記述子ロジックによって生成された記述子を保管する記述子テーブルとを含む装置。
【請求項２】
記述子テーブルが、第１データ処理システムで保管される、請求項１に記載の装置。
【請求項３】
記述子テーブルが、第２データ処理システムで保管される、請求項１に記載の装置。
【請求項４】
記述子ロジックが、記述子テーブル内の別の記述子へのリンクを含む分岐記述子を生成する、請求項１ないし３のいずれかに記載の装置。
【請求項５】
記述子テーブルが、その中の分岐記述子を介して一緒に順次リンクされる複数の記述子リストを含む、請求項４に記載の装置。
【請求項６】
記述子テーブルが、循環記述子リストを含む、請求項４に記載の装置。
【請求項７】
第１データ処理システムが、ホスト・コンピュータ・システムを含む、請求項１ないし６のいずれかに記載の装置。
【請求項８】
第２データ処理システムが、ホスト・コンピュータ・システムとデータ通信ネットワークとの間でデータを通信するデータ通信インターフェースを含む、請求項１ないし７のいずれかに記載の装置。
【請求項９】
メモリを有するホスト処理システムと、ホスト・コンピュータ・システムとデータ通信ネットワークとの間でデータを通信するデータ通信インターフェースと、ホスト・コンピュータ・システムのメモリとデータ通信インターフェースとの間でのデータのフローを制御する、請求項１ないし８のいずれかに記載の装置とを含むデータ処理システム。
【請求項１０】
メモリを介して第１および第２のデータ処理システムの間でデータのフローを制御する方法であって、記述子ロジックによって、メモリ内の位置と第２データ処理システムとの間で通信されるデータ・パケットを定義するフレーム記述子およびメモリ内の位置を識別するポインタ記述子を含む複数の記述子を生成することと、第１および第２のデータ処理システムによるアクセスのために、記述子ロジックによって生成された記述子を記述子テーブルに保管することとを含む方法。
【請求項１１】
記述子テーブルを第１データ処理システムに保管することを含む、請求項１０に記載の方法。
【請求項１２】
記述子テーブルを第２データ処理システムに保管することを含む、請求項１０に記載の方法。
【請求項１３】
記述子ロジックによって、記述子テーブル内の別の記述子へのリンクを含む分岐記述子を生成することを含む、請求項１０ないし１２のいずれかに記載の方法。
【請求項１４】
記述子テーブルを形成するために、複数の記述子リストを、分岐記述子を介して一緒に直

列にリンクすることを含む、請求項１３に記載の方法。
【請求項１５】
第１データ処理システムが、ホスト・コンピュータ・システムを含む、請求項１０ないし１４のいずれかに記載の方法。
【請求項１６】
第２データ処理システムが、ホスト・コンピュータ・システムとデータ通信ネットワークとの間でデータを通信するデータ通信インターフェースを含む、請求項１０ないし１５のいずれかに記載の方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、ホスト・コンピュータ・システムとデータ通信ネットワークの間でデータを通信するホスト・コンピュータ・システムおよびデータ通信インターフェースなどの第１および第２のデータ処理システムの間でメモリを介してデータのフローを制御する方法および装置に関する。
【背景技術】
【０００２】
従来のデータ処理ネットワークには、イーサネット（Ｒ）・アーキテクチャなどの中間のネットワーク・アーキテクチャによってすべてが相互接続される複数のホスト・コンピュータ・システムおよび複数の付加デバイスが含まれる。ネットワーク・アーキテクチャには、通常は、１つまたは複数のデータ通信交換機が含まれる。ホスト・コンピュータ・システムおよび付加デバイスのそれぞれが、データ処理ネットワーク内のノードを形成する。各ホスト・コンピュータ・システムには、通常は、ＰＣＩバス・アーキテクチャなどのバス・アーキテクチャによって相互接続される、複数の中央処理装置およびデータ・ストレージ・メモリ・デバイスが含まれる。ネットワーク・アダプタも、ホスト・コンピュータ・システムとデータ処理ネットワーク内の他のノードの間でネットワーク・アーキテクチャを介してデータを通信するためにバス・アーキテクチャに接続される。
【発明の開示】
【発明が解決しようとする課題】
【０００３】
ホスト・コンピュータ・システムとネットワーク・アーキテクチャの間のデータおよび制御情報の転送をできる限り効率的に促進することが望ましい。
【課題を解決するための手段】
【０００４】
本発明によれば、メモリを介して第１および第２のデータ処理システムの間でデータのフローを制御する装置であって、メモリ内の位置と第２データ処理システムの間で通信されるデータ・パケットを定義するフレーム記述子およびメモリ内の位置を識別するポインタ記述子を含む複数の記述子を生成する記述子ロジックと、第１および第２のデータ処理システムによるアクセスのために記述子ロジックによって生成された記述子を保管する記述子テーブルとを含む装置が提供される。
【０００５】
記述子ロジックおよび記述子テーブルによって、ホスト・コンピュータ・システムとデータ通信ネットワークの間でデータを通信するホスト・コンピュータ・システムとデータ通信インターフェースなどの第１および第２のデータ処理システムの間のデータ・フロー制御の効率が改善される。
【０００６】
記述子テーブルは、ホスト・コンピュータ・システムのメモリに保管することができる。代替案では、記述子テーブルが、データ通信インターフェースのメモリに保管される。記述子ロジックは、記述子テーブル内の別の記述子へのリンクを含む分岐記述子も生成することができる。記述子テーブルに、その中の分岐記述子を介して一緒に順次リンクされた

複数の記述子リストを含めることができる。代替案では、記述子テーブルに、循環記述子リストが含まれる。

【０００７】

本発明は、メモリを有するホスト処理システムと、ホスト・コンピュータ・システムとデータ通信ネットワークの間でデータを通信するデータ通信インターフェースと、ホスト・コンピュータ・システムのメモリとデータ通信インターフェースの間でのデータのフローを制御する、上に記載の装置とを含むデータ処理システムに拡張される。

【０００８】

もう１つの態様から本発明を見ると、メモリを介して第１および第２のデータ処理システムの間でデータのフローを制御する方法であって、記述子ロジックによって、メモリ内の位置と第２データ処理システムの間で通信されるデータ・パケットを定義するフレーム記述子およびメモリ内の位置を識別するポインタ記述子を含む複数の記述子を生成することと、第１および第２のデータ処理システムによるアクセスのために、記述子ロジックによって生成された記述子を記述子テーブルに保管することとを含む方法が提供される。

【０００９】

本発明の好ましい実施形態を、例のみとして、添付図面に関してこれから説明する。

【発明を実施するための最良の形態】

【００１０】

まず図１を参照すると、本発明を実施するデータ処理ネットワークの例に、InfiniBandネットワーク・アーキテクチャ（InfiniBandはInfiniBandTrade Association社の商標である）などの中間ネットワーク・アーキテクチャ３０によって相互接続された、複数のホスト・コンピュータ・システム１０および複数の付加デバイス２０が含まれる。ネットワーク・アーキテクチャ３０には、通常は、複数のデータ通信交換機４０が含まれる。ホスト・コンピュータ・システム１０および付加デバイス２０のそれぞれが、データ処理ネットワーク内のノードを形成する。各ホスト・コンピュータ・システム１０に、ＰＣＩバス・アーキテクチャなどのバス・アーキテクチャ７０によって相互接続される、複数の中央処理装置（ＣＰＵ）５０およびメモリ６０が含まれる。ネットワーク・アダプタ８０も、ホスト・コンピュータ・システム１０と、データ処理ネットワーク内の他のノードとの間で、ネットワーク・アーキテクチャ３０を介してデータを通信するために、バス・アーキテクチャに接続される。

【００１１】

図２を参照すると、本発明の特に好ましい実施形態では、ネットワーク・アダプタ８０に、ホスト・コンピュータ・システム１０のバス・アーキテクチャ７０への取外し可能な挿入のためのエッジ・コネクタなどのコネクタを有するプラグ可能オプション・カードが含まれる。オプション・カードは、コネクタ２７０を介してバス・アーキテクチャ７０に接続可能な特定用途向け集積回路（ＡＳＩＣ）またはインテグレーテッド・システム・オン・ア・チップ（Integrated System on a Chip、ＩＳＯＣ）１２０、ＩＳＯＣ１２０に接続される１つまたは複数の第３レベル・メモリ・モジュール２５０、および、ネットワーク・アーキテクチャ３０の媒体とＩＳＯＣ１２０の間でデータを通信するためにＩＳＯＣ１２０に接続されたインターポーザ２６０を担持する。インターポーザ２６０は、ネットワーク３０への物理的接続を提供する。本発明のいくつかの実施形態では、インターポーザ２６０を、単一のＡＳＩＣ内で実施することができる。しかし、本発明の他の実施形態では、インターポーザ２６０を、複数の構成要素によって実施することができる。たとえば、ネットワーク３０に光ネットワークが含まれる場合に、インターポーザ２６０に、別の光トランシーバを駆動するリタイマ（retimer）を含めることができる。メモリ２５０は、ＳＲＡＭ、ＳＤＲＡＭ、またはその組合せによって実施することができる。他の形態のメモリも、メモリ２５０の実施に使用することができる。ＩＳＯＣ１２０には、第１および第２のメモリが含まれる。アダプタ８０のメモリ・サブシステムを、すぐに説明する。以下の説明から明らかになるように、この配置は、データ処理ネットワークで動作する分散アプリケーションの改良された性能、改良されたシステム・スケーラビリティ、ある

範囲の通信プロトコルとの互換性、およびホスト・コンピュータ・システムでの減らされた処理要件を提供する。具体的に言うと、この配置では、アダプタ８０とホスト・システム１０の間の異種通信プロトコルの共存が可能になる。そのようなプロトコルは、さまざまなアプリケーションをサービスし、同一のアダプタ８０を使用し、データ構造の事前定義の組を使用することができ、これによって、ホストとアダプタ８０の間のデータ転送が機能強化される。並列にオープンすることができるアプリケーション・チャネルの数は、アダプタ８０に割り振られるメモリ・リソースの量によって決定され、アダプタに組み込まれる処理能力に依存しない。下記から、複数の構成要素を単一の集積回路チップに統合するというＩＳＯＣ１２０の概念によって、有利なことに、製造コストが最小になり、再利用可能なシステム基本構成要素が提供される。しかし、本発明の他の実施形態で、ＩＳＯＣ１２０の要素を、別個の構成要素によって実施できることも諒解されたい。

【００１２】

以下の説明では、用語「フレーム」が、ホスト・コンピュータ・システム１０で稼動するソフトウェアとアダプタ８０の間で転送されるデータ単位またはメッセージを指す。各フレームに、フレーム・ヘッダおよびデータ・ペイロードが含まれる。データ・ペイロードに、ユーザ・データ、高水準プロトコル・ヘッダ・データ、肯定応答、フロー制御、またはこれらの任意の組合せを含めることができる。フレーム・ヘッダの内容を、詳細にすぐに説明する。アダプタ８０は、フレーム・ヘッダだけを処理する。アダプタ８０は、フレームを、ネットワーク・アーキテクチャ３０でより効率的に移送される、より小さいパケットに断片化することができる。しかし、そのような断片化では、一般に、データ・ペイロードは変換されない。

【００１３】

本発明の特に好ましい実施形態では、データが、ネットワーク・アーキテクチャ３０上で、以下でパケットと称するアトミックな単位で移送される。各パケットには、経路情報と、その後のハードウェア・ヘッダ・データおよびペイロード・データが含まれる。本発明の通常の例では、１０２４バイトまでのパケット・サイズが使用される。より大きいサイズのフレームは、１０２４バイト・パケットに断片化される。本発明の他の実施形態で、異なるパケット・サイズを使用できることを諒解されたい。

【００１４】

本発明の好ましい実施形態では、アダプタ８０と、ホスト・コンピュータ・システム１０で稼動する複数のアプリケーションとの間の通信が、論理通信ポート（Logical Communication Port）アーキテクチャ（ＬＣＰ）を介してもたらされる。アダプタ８０には、異なる内部データ構造へのアクセス待ち時間の最適化を可能にするメモリ階層が含まれる。このメモリ階層を、すぐに説明する。本発明の好ましい実施形態では、アダプタ８０が、ネットワーク・アーキテクチャ３０に宛てられたアウトバウンド（ＴＸ）データとホスト・コンピュータ・システム１０に宛てられたインバウンド（ＲＸ）データに別々のパスを提供する。各パスには、それ自体のデータ転送エンジン、ヘッダ処理ロジック、およびネットワーク・アーキテクチャ・インターフェースが含まれる。これらのパスも、詳細にすぐに説明する。

【００１５】

図３を参照すると、ＬＣＰアーキテクチャによって、ホスト・コンピュータ・システム１０で稼動するローカル・コンシューマとアダプタ８０の間のインターフェースのフレームワークが定義される。そのようなコンシューマの例に、アプリケーションとスレッドの両方が含まれる。コンピュータ・システム１０を、ユーザ・アプリケーション空間９０とカーネル空間１１０に副分割することができる。ＬＣＰアーキテクチャは、各コンシューマにネットワーク・アーキテクチャ３０への論理ポートを与える。このポートには、ユーザ空間９０から直接にアクセスすることができる。本発明の特に好ましい実施形態では、ハードウェア保護機構が、アクセス許可を解決する。ＬＣＰ登録が、データ・フレームの転送の前に、カーネル空間１１０によって実行される。ＬＣＰアーキテクチャでは、通信プロトコルを定義する必要がない。そうではなく、ＬＣＰアーキテクチャでは、データおよ

び制御情報を転送するための、アプリケーションとアダプタ８０の間のインターフェースが定義される。その代わりに、通信プロトコルの詳細を、アプリケーションおよびアダプタ８０で実行されるプログラム・コードによってセットすることができる。アダプタ８０で使用できるチャネルの数は、ＬＣＰ関連情報に使用可能なアダプタ・カード８０上のメモリの量だけによって制限される。各ＬＣＰポートを、特定の特徴の組を有するようにプログラムすることができる。特徴の組は、特定のプロトコルに従って、ホスト・コンピュータ・システム内のメモリ６０とアダプタ８０の間でのデータ転送を最もよくサポートするように選択される。さまざまな通信プロトコルを、各プロトコルで異なるＬＣＰポートを使用して、同時にサポートすることができる。

【００１６】

ＬＣＰアーキテクチャには、ＬＣＰクライアント１００、カーネル空間１１０に常駐するＬＣＰマネージャ１３０、および、アダプタ８０に常駐する１つまたは複数のＬＣＰコンテキスト１４０が含まれる。

【００１７】

各ＬＣＰクライアント１００は、ＬＣＰポートに接続された、単一方向アプリケーション・エンド・ポイントである。ＬＣＰクライアント１００を、ユーザ・アプリケーション空間９０またはカーネル１１０内に配置することができる。動作中に、各ＬＣＰクライアント１００は、メモリ６０から読み取られ、アダプタ８０によってＴＸ　ＬＣＰチャネルを介して転送されるコマンドおよびデータを作るか、アダプタ８０によってメモリ６０へＲＸ　ＬＣＰチャネルを介して転送されるデータを消費する。

【００１８】

ＬＣＰマネージャ１３０は、ＬＣＰチャネルの割振りおよび割振り解除と、チャネルごとのメモリ６０内の読取／書込区域の登録の要求をサービスする、信頼される構成要素である。ＬＣＰマネージャ１３０は、他の通信動作、アプリケーション、またはホスト・コンピュータ・システム１０のオペレーティング・システムを危険にさらさずに、ユーザ空間アプリケーションが、アダプタ８０のリソースを使用できるようにする。

【００１９】

各ＬＣＰコンテキスト１４０は、特定のＬＣＰクライアント１００をサービスするのにアダプタ８０が必要とする制御情報の組である。ＬＣＰコンテキスト１４０には、可能なコマンド、ポインタ構造、およびバッファ記述子定義など、チャネルの存在全体を通じて一定のＬＣＰチャネル属性を含めることができる。ＬＣＰコンテキスト１４０には、サービスを待っているデータの量、関連するＬＣＰチャネルにアクセスするための次のアドレスなど、ＬＣＰチャネルに関する特定のＬＣＰサービス情報も含めることができる。ＬＣＰコンテキスト１４０は、アダプタ８０に常駐するメモリに保管されて、アダプタ８０があるチャネルのサービスを停止し、別のチャネルのサービスを開始する時の高速ＬＣＰコンテキスト切替が可能になっている。

【００２０】

ＬＣＰポートの開始を要求するＬＣＰクライアント１００は、ＬＣＰマネージャ１３０に頼り、ＬＣＰチャネルの割振りを要求する。ＬＣＰチャネル属性は、この時に決定され、これによって、ＬＣＰポートの挙動およびＬＣＰクライアント１００がＬＣＰポートに関連して実行を許可される動作が規定される。ＬＣＰクライアント１００は、一意の保護された形でアダプタ８０にアクセスするのに使用されるアドレスを許可される。このアドレスを、ドアベル（Doorbell）アドレスと称する。

【００２１】

ＬＣＰマネージャ１３０は、アダプタによる仮想アドレスから物理アドレスへの変換を可能にし、ユーザ空間クライアントが他のプログラムを改竄せずにこれらのホスト・メモリ区域にアクセスできるようにするために、ホスト・メモリ６０の区域を登録する責任も負う。

【００２２】

新しいバッファの登録および前のバッファの登録解除は、実行時中に各ＬＣＰクライアン

ト１００が要求することができる。そのような変更は、ＬＣＰクライアント１００、ＬＣＰマネージャ１３０、およびアダプタ８０の間の情報交換のシーケンスを必要とする。

【００２３】

各ＬＣＰクライアント１００およびポートは、ＬＣＰポートによってコマンド実行のために送られる保留中の要求をサービスするためにアダプタ８０が必要とするすべての情報を提供するＬＣＰコンテキスト１４０に関連する。

【００２４】

ＬＣＰクライアント１００とアダプタ８０の間のメモリ転送を開始し、フレームの送信を開始するために、ＬＣＰクライアント１００は、特定の動作に関する情報を保持する記述子を用意する。ＬＣＰクライアント１００は、アダプタ８０にマッピングされたドアベル・アドレスへの入出力書込を実行する。ドアベル・アドレスに書き込むことによって、アダプタ８０のＬＣＰコンテキスト１４０が更新され、新しい要求が実行のために追加される。

【００２５】

アダプタ８０は、保留中の要求を有するさまざまな送信ＬＣＰポートの間で調停し、次にサービスされる送信ＬＣＰポートを選択する。

【００２６】

データの受信時に、受信されたパケットのフレームおよびＬＣＰが、識別される。記述子を生成して、受信されたＬＣＰに必要な動作を定義する。アダプタ８０のＬＣＰエンジンによるこれらの記述子の実行によって、着信データが、ホスト・コンピュータ・システム１０のメモリ６０内でＬＣＰチャネルに割り振られた適当なデータ・バッファに保管される。

【００２７】

サービスされるＬＣＰチャネルごとに、アダプタ８０は、関連するＬＣＰコンテキスト情報をロードし、この情報を使用して、データ転送の所望の組を実行する。その後、アダプタ８０は、次に選択されるＬＣＰコンテキスト１４０の処理に継続する。

【００２８】

図４を参照すると、前に述べたように、ＩＳＯＣ１２０に、第１メモリ空間２２０および２３０と第２メモリ空間２４０が含まれ、アダプタ８０に、さらに、第３レベル・メモリ２５０が含まれる。第１、第２、および第３のメモリは、アダプタ８０のメモリ・サブシステム２１０のために間隔を空ける。本発明の好ましい実施形態では、ＩＳＯＣ１２０に、データ送信動作専用のＴＸプロセッサ（ＴＸ　ＭＰＣ）１５０と、データ受信動作専用のＲＸプロセッサ（ＲＸ　ＭＰＣ）１６０が含まれる。本発明の特に好ましい実施形態では、プロセッサ１５０および１６０が、ＩＢＭ　ＰｏｗｅｒＰＣ　４０５　ＲＩＳＣマイクロプロセッサなどの縮小命令セット・コンピューティング（ＲＩＳＣ）マイクロプロセッサによって実施される。メモリ・サブシステム２１０内で、ＩＳＯＣ１２０に、第１および第２のメモリ空間の他に、ＴＸプロセッサ１５０に関連するデータ・キャッシュ１８０および命令キャッシュ１７０が、ＲＸプロセッサ１６０に関連する第２データ・キャッシュ１９０および第２命令キャッシュ２００と共に含まれる。３つのレベルの間の相違は、メモリのサイズおよび関連するアクセス時間である。すぐに明らかになるように、メモリ・サブシステム２１０は、ＴＸプロセッサ１５０およびＲＸプロセッサ１６０の両方による命令およびデータへの便利なアクセスと、スケーラビリティと、削減された製造コストのためのＴＸプロセッサ１５０およびＲＸプロセッサ１６０の間のリソースの共用とを容易にする。

【００２９】

第１レベル・メモリ空間（Ｍ１）２２０および２３０に、ＴＸ－Ｍ１メモリ空間２２０およびＲＸ－Ｍ１メモリ空間２３０が含まれる。ＴＸ－Ｍ１メモリ２２０は、ＴＸプロセッサ１５０によってのみアクセスでき、ＲＸ－Ｍ１メモリ２３０は、ＲＸプロセッサ１６０によってのみアクセスできる。動作中に、第１レベル・メモリ空間２２０および２３０は、一時データ構造、ヘッダ・テンプレート、スタックなどを保持するのに使用される。第

１レベル・メモリ空間２２０および２３０の両方が、０待ち状態に反応する。第１レベル・メモリ空間２２０および２３０のそれぞれは、プロセッサ１５０および１６０の対応する１つのデータ・インターフェースだけに接続され、命令インターフェースには接続されない。この配置によって、キャッシュ可能およびキャッシュ不能の両方の第１レベル・メモリ区域を使用可能にすることができるのと同時に、第１レベル・メモリ空間２２０および２３０内のデータへの効率的なアクセスを維持できるようになる。

【００３０】

第２レベル・メモリ空間（Ｍ２）２４０は、プロセッサ１５０および１６０の両方、アダプタ８０の他の構成要素、およびホスト・コンピュータ・システム１０から使用可能な共用メモリである。第２レベル・メモリ空間２４０へのアクセスは、第１レベル・メモリ区域２２０および２３０へのアクセスより低速である。というのは、第２レベル・メモリ空間２４０が、共用される内部バスを介してより多くのエージェントによって使用されるからである。第３レベル・メモリ空間２５０も、共用リソースである。本発明の特に好ましい実施形態では、アダプタ８０に、その上で第１レベル・メモリ空間２２０および２３０と第２レベル・メモリ空間２４０の両方がプロセッサ１５０および１６０と同一のＡＳＩＣに集積される、コンピュータ周辺回路カードが含まれる。共用メモリ空間２４０および２５０は、一般に、高速で頻繁なアクセス・サイクルを必要としないデータ・タイプに使用される。そのようなデータ・タイプには、ＬＣＰコンテキスト１４０および仮想アドレス変換テーブルが含まれる。この共用メモリ空間２４０および２５０は、プロセッサ１５０および１６０の命令インターフェースおよびデータ・インターフェースの両方からアクセス可能である。

【００３１】

アダプタ８０は、送信データ・フローと受信データ・フローを別々に処理する。送信パスおよび受信パスのプロセッサ１５０および１６０は、タスク間の切替のオーバーヘッドを防止し、あるパス内の一時的な処理負荷を他のパスから分離し、着信データ・ストリームおよび発信データ・ストリームの処理に２つの組込みプロセッサを使用することを促進する。図５を参照すると、ＩＳＯＣ１２０に、送信パス・ロジック２８０および受信パス・ロジック２９０と、共用ロジック３００が含まれる。送信パス・ロジック２８０には、各ＬＣＰチャネルの詳細をデコードし、ＬＣＰ関連コマンドを実行のために取り出すＬＣＰ　ＴＸエンジン３１０と、アダプタ８０へのフレームの転送を制御するＴＸロジック３２０と、ＴＸフレームおよびパケット処理の管理用の前述のＴＸプロセッサ１５０と、命令および一時データ構造を保持する前述の第１レベルＴＸメモリ２２０と、リンク・ロジック３３０と、フレームのデータ・パケットへの断片化の経路指定処理などのデータ・フロー処理およびパケット処理を管理する際にＴＸプロセッサ１５０を支援するロジックとが含まれる。ＴＸプロセッサ１５０は、プロセッサが例外およびエラーの際にのみ割り込まれる、ポーリングのみ方式に基づいてタスクを直列に処理する。第１レベルＴＸメモリ２２０は、プロセッサ１５０によって、ＴＸロジック３２０との通信に使用される。受信パス・ロジック２９０には、リンク・ロジック３４０と、着信パケットのヘッダ処理およびそのようなパケットのフレームへの変換または組立の際に前述のＲＸプロセッサ１６０を支援するハードウェアと、ＲＸフレームおよびパケット処理用の前述のＲＸプロセッサ１６０と、命令を保持する前述の第１レベルＲＸメモリ２３０と、ネットワーク・アーキテクチャ３０からのフレームの転送を制御するＲＸロジック３５０と、各ＬＣＰチャネルの詳細をデコードし、関連するＬＣＰデータ構造内の着信データをホスト・コンピュータ・システムのメモリ６０に保管し、ＬＣＰクライアント１００によってアダプタ８０による使用のために供給される時に、空フレーム・バッファへのポインタを受け入れ、登録する、ＬＣＰ　ＲＸエンジン３６０とが含まれる。ＲＸプロセッサ１６０は、ＲＸプロセッサ１６０が例外およびエラーの際にのみ割り込まれる、ポーリングのみ方式を使用してタスクを直列に処理する。第１レベルＲＸメモリ２３０は、ＲＸプロセッサ１６０によって、ＲＸロジック３５０との通信に使用される。

【００３２】

前に述べたように、ＩＳＯＣ手法では、アダプタ８０および、回路ボードおよび他のサポート・モジュールなどのアダプタの他の構成要素に関連する製造コストの削減が可能になる。ＩＳＯＣ手法では、アダプタ８０の単純さも高まり、これによって信頼性が高まる。ＩＳＯＣ１２０の要素の間の接続の数は、効果的に無制限である。したがって、複数の幅広い相互接続パスを実施することができる。ホスト・コンピュータ・システム１０でのデータ処理オーバーヘッドを減らすために、ホスト・メモリ６０との間のデータ転送動作は、大部分がＩＳＯＣ１２０によって実行される。ＩＳＯＣ１２０は、着信パケットおよび発信パケットのヘッダの処理も実行する。送信中に、ＩＳＯＣ１２０は、ヘッダを作り、ネットワーク・アーキテクチャ３０に経路指定する。受信中に、アダプタ８０は、システムのメモリ内でのヘッダの位置を判定するためにヘッダを処理する。レベル１メモリ２２０および２３０は、０待ち状態メモリであり、スタック、テンプレート、テーブル、および一時保管場所などのプロセッサ・データ空間を提供する。本発明の特に好ましい実施形態では、送信パス・ロジック２８０、受信パス・ロジック２９０、および共用ロジック３００が、コアと称する、より小さい論理要素から作られる。用語コアが使用されるのは、これらの要素が、それらを異なる応用例に使用できるようにする独立型の特性を有する個々のロジックとして設計されるからである。

【００３３】

前に示したように、送信パス・ロジック２８０は、送信フレームまたは発信フレームを処理する責任を負う。フレーム送信は、パス・アーキテクチャ７０を介して、ホスト・コンピュータ・システム１０のＣＰＵ５０などのＣＰＵによって開始される。ＩＳＯＣ１２０には、パス・アーキテクチャ７０と通信するパス・インターフェース・ロジック３７０が含まれる。ＩＳＯＣ１２０には、パス・インターフェース・ロジック３７０をＩＳＯＣ１２０のプロセッサ・ローカル・パス（ＰＬＢ）３９０に接続するパス・プリッジング・ロジック３８０も含まれる。ＬＣＰ　ＴＸエンジン３１０は、コマンドおよびフレームをホスト・メモリ６０から取り出す。ＴＸプロセッサ１５０は、各フレームのヘッダを処理して、ネットワーク・アーキテクチャ３０上でパケットとして送信するのに適するフォーマットにする。ＴＸロジック３２０は、フレーム・データを修正なしで転送する。リンク・ロジック３３０は、送信される各パケットを処理して、ネットワーク・アーキテクチャ３０での送信用の最終的な形にする。リンク・ロジック３３０には、それぞれがネットワーク・アーキテクチャ３０に接続可能である１つまたは複数のポートを含めることができる。

【００３４】

前に示したように、受信パス・ロジック２９０は、着信パケットを処理する責任を負う。まず、ネットワーク・アーキテクチャ３０から受信されたパケットが、リンク・ロジック３４０によって処理される。リンク・ロジック３４０は、ヘッダおよびペイロードのフォーマットのパケットを再作成する。パケット・フォーマットおよびホスト・メモリ６０内での宛先を判定するために、ヘッダが、ＲＸプロセッサ１６０によって処理される。リンク・ロジック３４０には、それぞれがネットワーク・アーキテクチャ３０に接続可能である１つまたは複数のポートを含めることができる。ＲＸ　ＬＣＰエンジンは、パス・アーキテクチャ３７０を介してデータをホスト・メモリ６０に転送する責任を負う。

【００３５】

送信パス・ロジック２８０には、ＴＸ　ＬＣＰエンジン３１０とＴＸプロセッサ１５０の間のＨｅａｄｅｒＩｎ先入れ先出しメモリ（ＦＩＦＯ）４００が含まれる。受信パス・ロジックには、ＲＸプロセッサ１６０とＲＸ　ＬＣＰエンジン３６０の間のＨｅａｄｅｒＯｕｔ　ＦＩＦＯ４１０が含まれる。追加のＦＩＦＯおよびキューを、ＴＸロジック３２０内およびＲＸロジック３５０内に設けることができる。これらのＦＩＦＯおよびキューを、すぐに説明する。

【００３６】

共用ロジック３００には、送信パス・ロジック２８０および受信パス・ロジック２９０によって共用されるすべての要素が含まれる。この要素には、前述のパス・インターフェー

ス・ロジック３７０、バス・ブリッジング・ロジック３８０、ＰＬＢ３９０、第２レベル・メモリ２４０、およびリモート第３レベル・メモリ２５０へのアクセスを提供するコントローラ４２０が含まれる。バス・インターフェース・ロジック３７０は、バス・アーキテクチャ７０上でマスタおよびスレーブの両方として動作する。スレーブとして、バス・インターフェース・ロジックは、ＣＰＵ５０が、第２レベル・メモリ２４０に、コントローラ４２０を介して第３レベル・メモリ２５０に、および、ＩＳＯＣ１２０の構成レジスタおよび状況レジスタにもアクセスできるようにする。そのようなレジスタは、一般に、ＣＰＵ５０、ＴＸプロセッサ１５０、およびＲＸプロセッサ１６０によってアクセスすることができる。マスタとして、バス・インターフェース・ロジックは、ＴＸ　ＬＣＰエンジン３１０およびＲＸ　ＬＣＰエンジン３６０が、ホスト・コンピュータ・システム１０のメモリ６０にアクセスできるようにする。図５では、「Ｍ」がマスタ接続、「Ｓ」がスレーブ接続を示す。

【００３７】

図６を参照すると、ＩＳＯＣ１２０を通るパケットフローは、一般に対称である。言い換えると、フローの全般的な構造は、送信方向と受信方向の両方で類似する。ＩＳＯＣ１２０は、第１インターフェース・ロジック４４０と、第１制御ロジック４６０と、プロセッサ・ロジック４８０と、第２制御ロジック４７０と、第２インターフェース・ロジック４５０とを含むとみなすことができる。パケットは、下記の形で処理される。

【００３８】

Ａ．送信方向では、情報が、バス・アーキテクチャ７０から第１インターフェース・ロジックを介してＩＳＯＣ１２０に持ってこられる。受信方向では、情報が、ネットワーク・アーキテクチャ３０から第２インターフェース・ロジック４５０を介してＩＳＯＣ１２０に持ってこられる。

【００３９】

Ｂ．送信方向では、第１インターフェース・ロジック４４０を介してＩＳＯＣ１２０に持ってこられた情報が、第１制御ロジック４６０によって処理される。受信方向では、第２インターフェース・ロジック４５０を介してＩＳＯＣに持ってこられた情報が、第２制御ロジック４７０によって処理される。

【００４０】

Ｃ．送信方向では、フレーム・ヘッダが、第１制御ロジック４６０で発信フレームについて抽出され、プロセッサ・ロジック４８０によって処理される。プロセッサ・ロジック４８０は、フレーム・ヘッダに基づいて、第２制御ロジック４７０用の命令を生成する。発信フレームのペイロードは、第２制御ロジック４７０に渡される。受信方向では、フレーム・ヘッダが、第２制御ロジック４７０で着信フレームから抽出され、プロセッサ・ロジック４８０によって処理される。プロセッサ・ロジック４８０は、フレーム・ヘッダに基づいて、第１制御ロジック４６０用の命令を生成する。着信フレームのペイロードは、第１制御ロジック４６０に渡される。両方の方向で、プロセッサ４８０は、ペイロード・データを直接には処理しない。

【００４１】

Ｄ．送信方向では、第２制御ロジック４７０が、プロセッサ・ロジック４８０から受け取った命令に従って発信ペイロード・データをパッケージする。受信方向では、第１制御ロジック４６０が、プロセッサ・ロジック４８０から受け取った命令に従って着信ペイロード・データをパッケージする。

【００４２】

Ｅ．送信方向では、情報が、第２インターフェース・ロジック４５０からその宛先ヘネットワーク・アーキテクチャ３０を介して移動される。受信方向では、情報が、第１インターフェース・ロジックからその宛先へバス・アーキテクチャ７０を介して移動される。

【００４３】

ホスト・コンピュータ・システム１０で動作するソフトウェアへのインターフェースを、４３０に示す。同様に、プロセッサの入力および出力で動作するマイクロコードへのイン

ターフェースを、４９０および５００に示す。

【００４４】

図７を参照して、以下は、ＩＳＯＣ１２０を通る送信データ・フレームのフローの１例のより詳細な説明である。ＩＳＯＣ１２０は、ＩＳＯＣ１２０内の情報のさまざまなフォーマットに基づいて、ＬＣＰコンテキスト・ドメイン５１０、フレーム・ドメイン５２０、およびネットワーク・ドメイン５３０に分割することができる。ＴＸ　ＬＣＰエンジン３１０には、ＬＣＰ要求ＦＩＦＯ５５０、直接メモリ・アクセス（ＤＭＡ）ロジック５６０、フレーム・ロジック５８０、および前述のＬＣＰコンテキスト・ロジック１４０が含まれる。ＬＣＰ要求ＦＩＦＯ５５０、ＤＭＡロジック５６０、およびＬＣＰ　ＴＸコンテキスト・ロジック５９０は、ＬＣＰコンテキスト・ドメイン５１０に常駐する。フレーム・ロジック５８０は、フレーム・ドメイン５２０に常駐する。ＴＸロジック３２０、第１レベルＴＸメモリ空間２２０、およびＴＸプロセッサ１５０は、フレーム・ドメイン５２０とネットワーク・ドメイン５３０の境界にまたがる。ＴＸリンク・ロジック３３０は、ネットワーク・ドメイン５３０に常駐する。本発明の特に好ましい実施形態では、ＨｅａｄｅｒＩｎ　ＦＩＦＯ４００が、第１レベルＴＸメモリ空間２２０に一体化される。一般に、ホスト・コンピュータ・システム１０で実行されるアプリケーションが、フレームを作成する。フレームは、その後、アダプタ８０のＴＸ　ＬＣＰチャネルを使用して送信される。アプリケーションとアダプタ８０の間のハンドシェークは、ＬＣＰマネージャ１３０によって前に実行された初期化を前提とする。ＬＣＰサービス要求を追加するために、ＬＣＰクライアント１００が、アダプタ８０に、１つまたは複数の追加の送信フレームが実行の準備ができていることを知らせる。これは、ドアベルに制御ワードを書き込むことによって実行される。ドアベルのアドレスは、ＬＣＰポートに一意に関連し、他のプロセスによるアクセスから保護されるアドレスを使用して、書込動作が、バス・アーキテクチャ７０上での物理書込サイクルに変換される形で割り振られる。アダプタ８０は、書込動作を検出し、特定のＬＣＰクライアント１００の前の要求の項目を増分することによって、新しい要求をログに記録する。これは、関係するＬＣＰコンテキスト１４０の一部である。アダプタ８０のメモリ・サブシステム２１０内に保持される調停リストも、更新される。単純な例では、調停で、保留中の要求を有するすべての送信ＬＣＰチャネルの間で前述のＦＩＦＯ方式５５０が使用される。あるＬＣＰチャネルがサービスされている間に、次のＬＣＰチャネルが選択される。サービス・サイクルは、対応するＬＣＰコンテキストがＴＸ　ＬＣＰエンジン３１０にロードされる時に開始される。その後、ＬＣＰコンテキスト１４０にアクセスして、ＬＣＰチャネルをサービスするアトミック・オペレーションを導出し、そのような動作のパラメータを判定する。たとえば、そのようなアトミック・オペレーションは、ＬＣＰコンテキスト１４０に記録されたＬＣＰチャネル属性に基づくものとすることができる。完全なサービス・サイクルに、通常は、ＬＣＰクライアント１００によって作成される複数のアトミックな記述子をフェッチし、実行するためにアダプタ８０によって実行されるアクティビティの組が含まれる。ＴＸ　ＬＣＰチャネルの場合に、サービス・サイクルに、一般に、ホスト・メモリ６０からアダプタ８０のメモリ・サブシステム２１０に複数のフレームを読み取ることが含まれる。最後に、修正を必要とするすべてのＬＣＰコンテキスト情報（言い換えると、ＬＣＰサービス情報）が、アダプタ８０のメモリ・サブシステム２１０内で更新される。一般に、アダプタ８０によってＬＣＰサービス・サイクル内で最初に実行される動作は、次に処理される記述子を取り出すことである。

【００４５】

ＩＳＯＣ１２０による送信フレームの処理には、通常は下記のステップが含まれる。

【００４６】

Ａ．後続ＬＣＰポート・フレーム記述子の取出

次に取り出される記述子のアドレスは、ＬＣＰチャネルのコンテキスト１４０の一部として保管される。アダプタ８０は、ホスト・メモリ６０から記述子を読み取り、ＬＣＰチャネル属性に基づいてその記述子をデコードする。記述子によって、新しいフレーム・ヘッ

ダのサイズ、データ・ペイロードのサイズ、およびこれらの項目の位置が定義される。

【００４７】

Ｂ．仮想アドレスの物理アドレスへの変換

データ・バッファが、アプリケーション内で仮想メモリ・アドレスによって参照される場合に、そのアドレスは、アドレス変換という追加処理を受けなければならない。この場合に、アプリケーションによって使用される仮想アドレスが、アダプタ８０がホスト・メモリ６０にアクセスする間にアダプタ８０によって使用可能な物理アドレスに変換される。これは、ページ境界を超えることを監視し、ＬＣＰマネージャ１３０によってアダプタ８０のメモリ・サブシステム２１０に書き込まれる物理ページ位置情報を使用することによって行われる。仮想アドレスから物理アドレスへの変換処理は、記述子テーブルが、信頼されないＬＣＰクライアント１００によって作成される場合のセキュリティ手段としても働く。これによって、ホスト・メモリ６０の無関係な区域への許可されないアクセスが防止される。

【００４８】

Ｃ．フレーム・ヘッダの読取

物理アドレッシングを使用して、ＴＸフレームのヘッダおよびペイロード・データが、ホスト・メモリ６０内のバッファから読み取られる。その後、ヘッダが、ＴＸ　ＨｅａｄｅｒＩｎ　ＦＩＦＯ４００に保管される。ヘッダ取出が完了した時に、アダプタ８０は、ヘッダの処理をＴＸプロセッサ１５０によって開始できることを示す内部フラグをセットする。

【００４９】

Ｄ．フレーム・データの読取

ペイロード・データが、ホスト・メモリ６０から読み取られ、アダプタ８０によってデータＦＩＦＯ５７０に保管される。データＦＩＦＯ５７０は、図７では、ＴＸロジック３２０に常駐するものとして図示されている。しかし、データＦＩＦＯ５７０を、第１レベルＴＸメモリ空間２２０に一体化することもできる。データ読取トランザクションは、送信されるデータのすべてがアダプタ８０のメモリ・サブシステム２１０に保管されるまで継続する。読取動作の完了に続いて、状況指示が、ＬＣＰクライアント１００に返される。ヘッダがＨｅａｄｅｒＩｎ　ＦＩＦＯ４００に読み取られてすぐに、ヘッダの処理を開始できることに留意されたい。データ全体が読み取られるのを待つ必要はない。

【００５０】

Ｅ．フレーム・ヘッダの処理

ヘッダ処理は、ＴＸプロセッサ１５０によって実行される。ヘッダ処理は、プロトコル依存であり、ＬＣＰアーキテクチャの外部のプロトコル情報が用いられる。ＴＸプロセッサ１５０は、ＴＸプロトコル・ヘッダ・マイクロコードを実行し、プロトコルおよび経路指定初期化シーケンス中にアダプタ８０のメモリ・サブシステム２１０に既に保管された経路指定テーブルおよび他の関連情報にアクセスする。ＴＸプロセッサ１５０は、新しいヘッダがＨｅａｄｅｒＩｎ　ＦＩＦＯ４００内で待っていることの指示を受け取る時に、ヘッダ処理を開始する。ヘッダ処理では、ネットワーク・アーキテクチャ３０を介してパケットを送信するのに使用されるフォーマットであり、経路指定情報を含む、１つまたは複数のパケット・ヘッダが作られる。ペイロード・サイズが、ネットワーク・アーキテクチャ３０によって許容される最大パケット・サイズより大きい場合には、元のペイロード・データの連続するデータ・セグメントに関してそれぞれが使用される複数のパケット・ヘッダを生成して、ネットワーク・アーキテクチャ３０を介する通信用のパケットを形成することによって、ペイロードを断片化する。

【００５１】

Ｆ．送信用のパケット・ヘッダのキューイング

パケットのヘッダ・ワード数およびデータ・ワード数を定義するコマンドおよびパケット・ヘッダ自体が、ＴＸプロセッサ１５０によって、第１レベル・メモリ空間２２０内のＴＸ　ＨｅａｄｅｒＯｕｔ　ＦＩＦＯ５４０に書き込まれる。

【００５２】
Ｇ．送信用のパケット・ヘッダおよびパケット・データのマージ
ネットワーク・アーキテクチャ３０でのパケットの送信は、コマンドがＨｅａｄｅｒＯｕｔ　ＦＩＦＯ５４０内で準備ができており、データＦＩＦＯ５７０に、関連するパケットの送信を完了するのに十分なデータが含まれる時に、必ずトリガされる。巡回冗長検査（ＣＲＣ）を、各パケットのヘッダおよびデータに追加することができる。各完全なパケットが、ＴＸリンク・ロジック３３０を介してネットワーク・アーキテクチャ３０に転送される。
【００５３】
各フレームの送信処理は、すべてのフレーム・データが、１つまたは複数のパケットによってネットワーク・アーキテクチャ３０で送信された時に完了する。アダプタ８０によって処理されるフレームごとに、第２のＬＣＰクライアント１００を介してアプリケーションに状況を返すことができる。この状況は、ホスト・メモリ６０からアダプタ８０へのフレーム・データ転送の完了、フレーム送信自体の完了、または他のレベルの送信状況とすることができる。
【００５４】
どの瞬間でも、アダプタ８０は、次にサービスされるＬＣＰの選択、ＬＣＰチャネルＡのサービスの開始、ＬＣＰチャネルＢの最後のフレームのデータのＤＭＡ取出の実行、ＬＣＰチャネルＣのフレーム・ヘッダの処理および断片化、ＬＣＰチャネルＤから発するパケットの送信のうちの一部またはすべてを平行に実行することができる。
【００５５】
図８を参照して、以下は、例のみとして、ＲＸ　ＬＣＰポートを使用するアプリケーションによるデータ・フレーム受信の説明である。ＩＳＯＣ１２０の動作は、ＬＣＰによってサポートされるプロトコルのタイプに応じて変更することができる。アプリケーションとアダプタ８０の間のハンドシェークは、ＬＣＰマネージャ１３０によって前に実行された初期化を前提とする。ＲＸ　ＬＣＰエンジン３６０には、ＬＣＰ割振りロジック６２０、ＬＣＰコンテキスト・ロジック６１０、およびＤＭＡロジック６３０が含まれ、これらのすべてが、ＬＣＰコンテキスト・ドメイン５１０に常駐する。ＲＸプロセッサ１６０、第１レベルＲＸメモリ空間２３０、およびＲＸロジック３５０のすべてが、フレーム・ドメイン５２０とネットワーク・ドメイン５３０の間の境界にまたがる。ＲＸリンク・ロジック３４０およびパケット援助ロジック６００は、ネットワーク・ドメイン５３０に常駐する。本発明の特に好ましい実施形態では、ＨｅａｄｅｒＯｕｔ　ＦＩＦＯ４１０が、第１レベルＲＸメモリ空間２３０内に配置される。ＩＳＯＣ１２０によってネットワーク・アーキテクチャ３０から受信されたフレームは、ホスト・メモリ６０内のＬＣＰクライアント・バッファに書き込まれる。メモリ・バッファの可用性は、ＬＣＰ　ＲＸクライアント１００によって判定され、着信データ・フレームの挿入のためにアダプタ８０に示される。ＬＣＰクライアント１００は、送信される準備ができた新しいフレームについて送信パス・ロジック２８０が知らされる前に述べた形に似て、ＩＳＯＣ１２０の受信ドアベルに書き込むことによってバッファを提供する。ドアベル・レジスタ・アドレスは、書込動作がパス・アーキテクチャ７０への物理書込サイクルに変換されるように割り振られる。アダプタ８０は、書込動作を検出し、特定のＬＣＰ　ＲＸクライアント１００の使用可能なワード項目の数を増分することによって、空のメモリ区域の新たな提供をログに記録する。使用可能なワード・カウントは、関係するＬＣＰコンテキスト１４０の一部である。アプリケーションは、バッファ内の受信したフレームの処理を完了する時に、必ずドアベルに書き込む。この書込サイクルによって、新たに使用可能なメモリ空間のワード数が示される。ＬＣＰコンテキスト内のカウントが、その量だけ増分される。ネットワーク・アーキテクチャ３０から受信されたパケットは、アダプタ８０によってホスト・メモリ６０内の連続する空間に組み立てられる、より大きいフレームの一部である場合がある。ＩＳＯＣ１２０による受信されたフレームの処理には、一般に、下記のステップが含まれる。
【００５６】

Ａ．パケット・ヘッダとデータの分離
ＲＸリンク・ロジック３４０は、ネットワーク・アーキテクチャ３０からの情報をパケットのストリームに変換する。受信されたパケットのそれぞれが、ＲＸリンク・ロジック３４０によって処理されて、パケット・ヘッダがペイロード・データから分離される。ヘッダは、第１レベルＲＸメモリ空間２３０内のＲＸ　ＨｅａｄｅｒＩｎ　ＦＩＦＯ６４０にプッシュされる。ペイロードは、ＲＸロジック３５０内のＲＸデータＦＩＦＯ６５０にプッシュされる。ＲＸデータＦＩＦＯ６５０は、第１レベルＲＸメモリ空間２３０内で実施することもできる。
【００５７】
Ｂ．パケット・ヘッダのデコードおよびＬＣＰフレーム・ヘッダの生成。
パケット・ヘッダをデコードして、パケットが属するフレームのＩＤ、ペイロードのサイズ、およびフレーム・データのサイズを示すフィールドを提供する。パケット・ヘッダが、ＲＸ　ＨｅａｄｅｒＩｎ　ＦＩＦＯ６４０に関するリーダーになるならば、指示が、ＲＸプロセッサ１６０に送られる。ＲＸプロセッサは、パケット・ヘッダ情報を処理し、パケット・データの転送に必要な情報を含むＬＣＰ関連コマンドを生成する。そのような情報には、パケットのアドレスおよび長さが含まれる。ヘッダ処理の終りに、記述子または記述子の組が、ＬＣＰ　ＲＸ　ＨｅａｄｅｒＯｕｔ　ＦＩＦＯ４１０に書き込まれ、指示がトリガされる。
【００５８】
Ｃ．ＲＸ　ＬＣＰコンテキスト内のデータの転送
記述子が、ＲＸ　ＬＣＰエンジン３６０によってＲＸ　ＨｅａｄｅｒＯｕｔ　ＦＩＦＯ４１０から取り出され、デコードされる。記述子には、ＬＣＰ番号、パケット・アドレス、パケット・データ長、および、アダプタ８０のメモリ・サブシステム２１０に転送されるデータのソース・アドレスが含まれる。ＲＸ　ＬＣＰエンジン３４０は、ＬＣＰコンテキスト情報を使用して、ホスト・メモリ６０内の書き込まれるターゲット物理アドレス（または、ページにまたがる場合には複数のアドレス）を作成し、ＤＭＡ転送を開始して、データを書き込む。
【００５９】
Ｄ．ＩＳＯＣ　ＤＭＡトランザクション
ＩＳＯＣ１２０は、適当なバス・コマンドを選択し、可能な最長のバーストを実行することによって、バス・アーキテクチャ７０でのトランザクションの最適化を目指す。
【００６０】
どの瞬間でも、アダプタ８０は、ＬＣＰチャネルＸのバッファ割振りの処理、ＬＣＰチャネルＡのインバウンド・データ書込サービスの開始、ＬＣＰチャネルＢのデータのＤＭＡストアの実行、ＬＣＰチャネルＣ宛のパケットのフレーム組立の処理、およびＬＣＰチャネルＤのパケットの受信の一部またはすべてを並列に実行することができる。
【００６１】
ＲＸプロセッサ１６０およびＴＸプロセッサ１５０でのフレーム処理オーバーヘッドを最小にするために、パケット援助ロジック６００に、フレーム断片化ロジック、ＣＲＣおよびチェックサム計算ロジック、およびマルチキャスト処理ロジックが含まれる。
【００６２】
ＴＸ　ＬＣＰエンジン３１０およびＲＸ　ＬＣＰエンジン３６０の両方とホスト１０の間のデータ・フローを、これから詳細に説明する。ＴＸ　ＬＣＰポートおよびＲＸ　ＬＣＰポートの両方で、データ転送用のメモリ・バッファと、そのようなメモリ・バッファをポイントする記述子構造が使用される。記述子構造は、データ・プロバイダとデータ・コンシューマの間でデータ・バッファを管理し、データ・プロバイダによって使用される空のメモリ・バッファを返すのに使用される。記述子は、物理アドレスまたは仮想アドレスのいずれかに基づいてメモリ・バッファをポイントする。
【００６３】
ＴＸ　ＬＣＰチャネルは、ホスト・メモリ６０からＩＳＯＣ１２０のバッファへのデータ

転送の責任を負う。ロジックの他の層は、ＩＳＯＣ１２０のバッファからネットワーク３０にデータを転送する責任を負う。ＲＸ　ＬＣＰチャネルは、ネットワーク３０から受信されたデータのホスト・メモリ６０への転送の責任を負う。

【００６４】

ＴＸ　ＬＣＰエンジン３１０およびＲＸ　ＬＣＰエンジン３６０は、比較的多数のＬＣＰチャネルを扱うことができる。各ＬＣＰチャネルは、それに固有のすべての情報を含むパラメータの組を有する。この情報には、チャネルの構成、現在の状態、および状況が含まれる。あるチャネルに関連するＬＣＰコンテキスト１４０が、チャネルの初期化中にＬＣＰマネージャ１３０によってセットされる。チャネル動作中に、ＬＣＰコンテキスト１４０の内容が、ＩＳＯＣ１２０のみによって更新される。ＬＣＰコンテキスト１４０は、アダプタ８０のメモリ・サブシステム２１０内のコンテキスト・テーブルに保管される。ＬＣＰチャネルのＬＣＰコンテキスト１４０へのアクセスは、ＬＣＰ番号に従って実行される。ＬＣＰのＲＸチャネルおよびＴＸチャネルでは、異なるＬＣＰコンテキスト構造が使用される。

【００６５】

データ・バッファは、ホスト１０のメモリ６０内のピン止めされた区域である。送信バッファには、送信用のデータが保持される。ＴＸ　ＬＣＰエンジン３１０は、これらのバッファに置かれたデータをＩＳＯＣ１２０の内部バッファに移動する。ネットワーク・アーキテクチャ３０から受信される着信データは、ＲＸ　ＬＣＰエンジン３６０によって、ホスト１０のメモリ６０内のバッファに移動される。バッファの所有権は、ホスト１０内のソフトウェアとＩＳＯＣ１２０の間で交番する。ＬＣＰ　ＴＸチャネルでのイベントの順序は、次の通りである。

Ａ．ホスト１０のソフトウェアが、ホスト１０のメモリ６０内で、送信されるデータと共にバッファを準備する。

Ｂ．ソフトウェアが、バッファ内のデータが送信の準備ができていることをＩＳＯＣ１２０に通知する。

Ｃ．ＩＳＯＣ１２０が、バッファからデータを読み取る。

Ｄ．ＩＳＯＣ１２０が、ホスト１０のソフトウェアに対して、読み取られ、新しいデータを転送するためにホスト１０のソフトウェアによって再利用することができるバッファを識別する。

【００６６】

ＬＣＰ　ＲＸチャネルでのイベントの順序は、次の通りである。

Ａ．ホスト１０のソフトウェアが、ＩＳＯＣ１２０が受信されたデータを書き込むことができるバッファを準備する。

Ｂ．ソフトウェアが、空きバッファがホストのメモリ６０内で準備ができていることをＩＳＯＣ１２０に通知する。

Ｃ．ＩＳＯＣ１２０が、データをバッファに書き込む。

Ｄ．ＩＳＯＣ１２０が、ホスト１０のソフトウェアに対して、受信されたデータが書き込まれ、ソフトウェアによって処理されることができるバッファを識別する。

【００６７】

ソフトウェアが、ＩＳＯＣ１２０によって使用されるバッファを準備する時に、バッファ情報が、ドアベル・レジスタを介して追跡される。ＩＳＯＣ１２０によって使用されるバッファに関する情報は、状況更新を使用してまたは完了キューを介して、ソフトウェアに返される。ＴＸ　ＬＣＰチャネルについて、バッファに、ＴＸ　ＬＣＰエンジン３１０によってＩＳＯＣ１２０に転送され、処理されて、ネットワーク３０での送信用の１つまたは複数のパケットになるデータおよびヘッダの情報が含まれる。ヘッダは、ＩＳＯＣ１２０のＴＸプロセッサ１５０によって、ネットワーク３０で送信されるパケットのヘッダを生成するのに使用される。ＲＸ　ＬＣＰチャネルについて、空きバッファが、ホスト１０のソフトウェアによってアダプタ８０に割り当てられる。アダプタ８０は、受信されたパケットをバッファに書き込む。

【００６８】
記述子によって、ＩＳＯＣ１２０およびホスト１０のソフトウェアの両方に既知のデータ構造が定義されている。ソフトウェアは、記述子を使用して、ＩＳＯＣ１２０に制御情報を転送する。制御情報は、所望の機能に応じて、フレーム記述子、ポインタ記述子、または分岐記述子の形とすることができる。ソフトウェア内およびＩＳＯＣ１２０内の記述子ロジックは、行われる制御手段に従って記述子を生成し、修正する。そのような手段を、すぐに説明する。フレーム記述子には、パケットの記述子（たとえば、データ長、ヘッダ長など）が含まれる。ポインタ記述子には、データ位置の記述子が含まれる。分岐記述子には、記述子位置の記述（たとえば、記述子のリンク・リスト）が含まれる。記述子内の情報は、ＴＸ　ＬＣＰエンジン３１０およびＲＸ　ＬＣＰエンジン３６０によって実行されるデータ移動動作の、ホスト１０内のソフトウェアによる制御に使用される。ＴＸパケット・ヘッダを生成するためにフレームを処理するのに使用される情報は、フレームのヘッダに配置される。図９を参照すると、記述子を、単一のテーブル７００内に設けることができ、ＬＣＰコンテキスト１４０が、テーブル７００の先頭をポイントする。図１０を参照すると、記述子を、リンクされた記述子テーブル７２０から７４０の構造で配置することもできる。ＬＣＰチャネル初期化の後に、ＬＣＰコンテキスト１４０は、構造内の最初の記述子テーブル７２０の先頭をポイントする。分岐記述子７５０から７７０は、テーブル７２０から７４０のリンク・リストを生成するのに使用され、ここで、記述子テーブル７２０から７４０の終りの分岐記述子７５０から７７０は、別のテーブル７２０から７４０の始めをポイントする。図９に戻ると、分岐記述子は、循環バッファを生成するのにも使用することができ、ここで、テーブル７００の終りの分岐記述子７１０は、同一のテーブル７００の始めをポイントする。循環バッファは、受信パスで使用することもできる。この場合に、ＬＣＰコンテキスト１４０が、バッファの先頭をポイントするように開始される。バッファは、ＩＳＯＣ１２０がその終りに達した時に、ラップ・アラウンドされる。ホスト１０のソフトウェアは、ホスト１０のメモリ６０に（受信パスと送信パスの両方について）、またはアダプタ８０のメモリ２５０に（送信パスのみについて）記述子を書き込むことができる。アダプタ８０のメモリ・サブシステム２１０への記述子の書込には、ホスト１０のソフトウェアによる入出力動作が含まれ、アダプタ８０のメモリ・サブシステム２１０が占有される。ホスト１０のメモリ６０での記述子の書込は、アダプタ８０が、新しい記述子を読み取らなければならない時に、必ずホスト１０のメモリ６０にアクセスすることを必要とする。ソフトウェア記述子の位置は、ＬＣＰチャネルごとに独立に、ＬＣＰマネージャ１３０によって定義される。記述子の位置は、システム性能最適化に従って定義される。記述子は、キューの構成での柔軟性を提供する。

【００６９】
ＴＸ　ＬＣＰエンジン３１０およびＲＸ　ＬＣＰエンジン３６０は、記述子テーブル内の記述子にアクセスし、データ・バッファにアクセスするのに、アドレスを使用する。アドレスは、物理アドレスまたは仮想アドレスのいずれかとすることができる。用語物理アドレスは、ＩＳＯＣ１２０が、そのままでバス７０に駆動できるアドレスを指す。用語仮想アドレスは、物理アドレスではなく、ソフトウェアまたはマイクロコードによって使用されるアドレスを指す。仮想アドレスは、物理アドレスを生成するために、マッピングを受けなければならない。ＴＸ　ＬＣＰエンジン３１０およびＲＸ　ＬＣＰエンジン３６０によって使用されるアドレスは、ＬＣＰチャネル・コンテキスト１４０内のポインタ、ホスト１０で稼動するソフトウェアによって準備される記述子内のポインタ、ＲＸプロセッサ１６０によって準備される記述子内のポインタ、およびＴＸプロセッサ１５０によって準備される記述子内のポインタ（完了メッセージを返すのに使用される）という異なるソースを有することができる。ポインタは、記述子またはデータ・バッファをポイントすることができる。ＴＸ　ＬＣＰエンジン３１０およびＲＸ　ＬＣＰエンジン３６０によって使用されるすべてのアドレスを、任意選択として、バス７０上の物理アドレスとして使用される新しいアドレスにマッピングすることができる。アドレス・マッピングは、ＴＸ　ＬＣＰエンジン３１０およびＲＸ　ＬＣＰエンジン３６０によって行われる。ＩＳＯＣ１２

０は、ローカル・メモリ２１０を使用して、変換テーブルを保持する。ＬＣＰマネージャ１３０は、メモリ登録中にアダプタ８０に変換テーブルを書き込む。アドレス・マッピングによって、バッファまたは記述子テーブルに仮想アドレスを使用できるようになる。仮想アドレッシングによって、物理的に複数の物理ページに配置される仮想バッファの管理が可能になる。アドレス・マッピングによって、ホスト１０が、ソフトウェアのための変換プロセッサを必要とせずに、仮想アドレスを使用するアプリケーションを直接に扱えるようになる。

【００７０】

図１１を参照すると、この図には、ホスト１０のソフトウェアから見えるバッファ８８０のイメージ８００が示されている。ホスト１０内のメモリ６０にアクセスするのに使用される、アドレスの物理マッピング８１０も示されている。仮想ポインタが、バッファ内の位置を８２０ポイントする。この例のバッファは、ホスト１０のメモリ６０内の少数の連続しないページ８４０から８７０を占める仮想バッファである。ＬＣＰエンジン３１０および３６０は、変換テーブル８３０を介してアドレスを変換することによってマッピングを実行する。変換テーブルは、仮想バッファ８８０からマッピングされる各物理バッファ８４０から８７０の先頭への物理アドレス・ポインタを保持する。アダプタ８０内でのアドレス・マッピングによって、ホスト１０のメモリ６０内で記述子およびデータ・バッファをマッピングする時の柔軟性が可能になる。アダプタ８０内でのアドレス・マッピングによって、ホスト１０のソフトウェアが物理アドレスへのアドレス変換を実行することを必要とせずに、仮想アドレスを使用するソフトウェア・バッファへの直接接続も可能になる。

【００７１】

アダプタ８０がホストのメモリ６０に書き込む各パケットは、それに関連する状況を有する。状況を用いると、アダプタ８０とホスト１０のソフトウェアの間の同期化が可能になる。状況は、パケットの異なる信頼性レベルを示すのに使用することができる。ＩＳＯＣ１２０は、下記の状況ライト・バックを提供する：送信ＤＭＡ完了は、ＴＸパケット内のデータがアダプタ８０に読み込まれたことを示し；信頼される送信は、ネットワーク３０でのデータ送信の完了を示すために返され；受信ＤＭＡ完了は、メモリ６０への受信データ転送の完了を示し；信頼される受信は、ネットワーク３０内の宛先ノードによる送信パケットの受信を示す。

【００７２】

ＴＸフレーム記述子には、２バイトの状況フィールドが含まれる。状況ライト・バックは、トランザクション状態が記述子にライト・バックされることを意味する。状況には、ホスト１０のソフトウェアがポーリングできる完了ビットが含まれる。ホスト１０のソフトウェアは、セットされた完了ビットを見つけた時に、そのフレーム記述子によって定義されるフレームに関連するバッファを再利用することができる。

【００７３】

完了キューは、ＲＸ　ＬＣＰチャネルによって実施される。完了キューによって使用されるＬＣＰチャネルは、どのＲＸ　ＬＣＰチャネルによっても実施することができる柔軟性および特性のすべてを有する。ＴＸプロセッサ１５０およびＲＸプロセッサ１６０は、状況ライト・バックを生成して、信頼される送信、信頼される受信、受信ＤＭＡ完了、または送信ＤＭＡ完了を示す。フレームに関連する異なる指示を、異なる場合に使用することができる。たとえば、信頼される送信の場合に、ＴＸプロセッサ１５０は、パケット送信の状況を示す内部レジスタを読み取る。信頼される受信の場合に、ＲＸプロセッサ１６０が、肯定応答を含む受信されたパケットとして完了指示を得る。受信ＤＭＡ完了の場合に、ＲＸプロセッサ１６０が、フレーム完了情報を使用する。送信ＤＭＡ完了の場合に、ＴＸプロセッサ１５０が、アダプタ８０での送信用フレームの受信を示す。完了キューは、単一のＴＸ　ＬＣＰチャネルまたは単一のＲＸ　ＬＣＰチャネルによって使用することができ、あるいは、複数のチャネルによって共用することができる。アダプタ８０のマイクロ・コードが、ＲＸ　ＬＣＰエンジン３６０のコマンド・キューへのフレーム記述子を開

始することによって、状況キューを更新する。図１２を参照すると、状況は、完了キュー９２０を含む完了状況ＬＣＰ９００を介してホスト１０のメモリ６０に転送される。完了キュー９２０は、連続し（物理的にまたは仮想的にのいずれか）、ホスト１０のメモリ６０内に配置される。たとえば、完了キューを、連続するバッファ内に保持することができる。完了キューの項目９３０は、固定されたサイズを有することが好ましい。各項目が、受信ＬＣＰ９１０に関連するバッファ９５０の先頭へのポインタ９４０を保持する。バッファ９５０には、完了状況に関連するパケット９６０が書き込まれる。

【００７４】

ＴＸソフトウェア／アダプタ・ハンドシェークに、ＴＸ　ＬＣＰポートよび完了ＲＸ　ＬＣＰポートが含まれる。各ＬＣＰ送信チャネルでは、下記のデータ構造が使用される。

メモリ・マップされたアドレスとして実施され、記述子およびデータを処理する増分要求についてアダプタ８０に知らせる、ドアベル項目。各プロセスは、ドアベル・アクセスに使用されるメモリ・マッピングされたアドレスの単一のページへの一意のアクセスを有する。

ＬＣＰ属性フィールドおよび状況フィールドを含む、アダプタ・メモリ空間２１０内のＬＣＰコンテキスト項目。

送信記述子の構造。この構造は、ホスト１０のメモリ６０内の複数の物理ページにまたがることができる。仮想アドレッシングが、記述子について使用される場合に、変換テーブルを使用して、あるページから次のページに移動する。物理アドレッシングが、記述子について使用される場合に、分岐記述子が、あるページから次のページへの移動に使用される。送信記述子には、アダプタ８０への記述子関連データのすべての転送の後に更新することができる状況フィールドが含まれる。

ポインタ記述子によってポイントされる、ホスト１０のメモリ６０内でピン止めされた送信データ・バッファ。仮想アドレッシングが、データ・バッファについて使用される場合に、変換テーブルによって、ポインタが、アダプタ８０によってホスト１０のメモリ６０にアクセスするのに使用される物理アドレスに変換される。

アダプタ・メモリ空間２１０内の変換テーブルおよび保護ブロックが、アドレス・マッピングに使用される。

【００７５】

図１３を参照すると、送信パケットのフローに、ステップ１０００で、ホスト１０のソフトウェア１０２０が、送信されるデータをバッファ１０３０に書き込むことが含まれる。ステップ１０１０で、ソフトウェア１０２０が、記述子１０４０を更新する。記述子１０４０は、ホスト１０のメモリ６０内またはアダプタ８０のメモリ・サブシステム２１０内のいずれかとすることができる。ステップ１０５０で、ソフトウェア１０２０が、ドアベルを鳴らして、新しいデータの送信の準備ができていることをアダプタ８０に通知する。ステップ１０６０で、アダプタ８０が、異なるＬＣＰチャネルからの要求の間の調停を管理する。あるチャネルが調停に勝った時に、アダプタ８０が、新しい記述子１０４０を読み取る。ステップ１０７０で、アダプタ８０が、データを読み取る。ステップ１０８０で、データが、ネットワーク３０に送信される。ステップ１０９０で、状況が、記述子１０４０内または完了キュー内で更新される。

【００７６】

ＴＸ　ＬＣＰチャネルでは、データ・バッファにアクセスする時にアドレス変換を使用することができる。この場合に、データ・バッファは、複数のメモリ・ページからなる。この処理に関する限り、これらのメモリ・ページは、連続する仮想メモリ空間内にある。しかし、アダプタ８０に関する限り、これらのメモリ・ページは、連続しない物理メモリ空間内にある場合がある。完了状況構造に、送信されたフレームの状況を示す情報が含まれる。これは、別々のＬＣＰチャネルとして実施される。すべてのフレームの最初の記述子であるフレーム記述子は、フレームがアダプタ８０に転送された後に更新することができる任意選択の状況フィールドを有する。

【００７７】

図１４を参照すると、送信ＬＣＰチャネル・フローの例で、記述子１１００が、ホスト１０のメモリ６０に配置される。記述子１１００およびパケット１１２０を保管するバッファ１１１０へのアクセスは、アダプタ８０に配置される変換テーブル１１３０を介するアドレス変換を必要とする。バッファ１１１０によって、ホスト１０のソフトウェアの仮想アドレス空間内の連続するスペースが使用される。各フレーム１１２０は、２タイプの記述子すなわち、パケットに関係する情報を与えるフレーム記述子１１４０と、データ１１２０を保持するバッファ１１１０をポイントするポインタ記述子１１５０とによって記述される。各パケットに、データ・ペイロード１１７０と、同一のバッファ１１８０でそれに先行するヘッダ１１６０が含まれる。

【００７８】

ドアベルへの書込トランザクション１１９０によって、アダプタ８０による使用に使用可能なワード数１２００が更新される。この情報は、ＬＣＰコンテキスト１４０に保管される。送信ＬＣＰコンテキスト１４０には、送信されるデータを保持するバッファ１１１０の先頭へのポインタ１２１０が含まれる。ＬＣＰチャネルが、ＩＳＯＣ１２０の内部調停に勝つ時に、ＩＳＯＣ１２０は、ＬＣＰコンテキスト１４０内のポインタ１２１０に従ってＬＣＰチャネルの記述子を読み取る。ＬＣＰチャネルの記述子１１００およびバッファ１１１０の両方に関する仮想アドレスは、アダプタ８０のメモリ・サブシステム２１０内に配置される変換テーブル１１３０を使用して物理アドレスに変換される。変換テーブル１１３０は、メモリ・バッファの登録中にＬＣＰマネージャ１３０によって更新される。ＩＳＯＣ１２０は、データおよびフレーム・ヘッダをバッファ１１１０からアダプタ８０へ読み取る。フレーム・ヘッダ１１６０が、ＩＳＯＣ上でネットワーク３０のヘッダに置換される１３２０。パケット・ヘッダおよび対応するデータが、ネットワーク３０に送信される。

【００７９】

ＲＸ　ＬＣＰポートは、着信データをＩＳＯＣ１２０からホスト１０で稼動するソフトウェア・アプリケーションによって使用されるメモリ６０に転送するのに使用される。ＴＸ　ＬＣＰチャネルは、完全に、ホスト１０のソフトウェアによって開始される記述子を介して制御される。ＲＸ　ＬＣＰチャネルでは、ホスト１０のソフトウェアおよびＩＳＯＣ１２０の両方からの記述子が使用される。ＩＳＯＣ１２０によって開始される記述子は、ＬＣＰチャネル動作を制御して、ホスト１０のメモリ６０内の受信されたフレームの宛先を定義するのに使用される。ホスト１０のソフトウェアによって開始される記述子は、バッファが変換テーブル内のマッピングを介して定義されない場合に、バッファの位置を定義するのに使用することができる。ホスト１０のソフトウェアとアダプタ８０の間のハンドシェークを実施するために、２つのＬＣＰチャネルすなわち、受信された着信データ構造を扱うＲＸ　ＬＣＰチャネルと、完了状況キューを扱うＲＸ　ＬＣＰチャネルが使用されることが好ましい。この完了状況は、アダプタ８０が、ホスト１０のソフトウェアに、ホスト１０のメモリ６０へのフレーム転送が完了したことをシグナリングするのに使用される。項目が、完了キュー構造の順次アドレスに挿入される。完了状況項目のそれぞれに、アダプタ８０によってマークされ、項目所有権がアダプタ８０からホスト１０のソフトウェアに転送されたことを検査するためにホスト１０のソフトウェアによってポーリングされるフィールドが含まれる。１つまたは複数のＲＸ　ＬＣＰチャネルが、同一の完了状況キューを使用することができる。複数のＲＸ　ＬＣＰチャネルによる完了状況キューの共用は、ＩＳＯＣ１２０によって実行される。

【００８０】

ＲＸ　ＬＣＰチャネルは、着信パケットの宛先アドレスを示す情報を必要とする。ＩＳＯＣ１２０は、空きバッファの位置を見つけるための下記の２つのアドレッシングを有する。

直接アドレッシング・モードは、バッファをポイントするのにポインタ記述子を使用しないＬＣＰチャネルを指す。宛先アドレスは、ＩＳＯＣ１２０のマイクロコードによって定義されるか、コンテキスト１４０から読み取られる。

間接アドレッシング・モードは、記述子構造内でデータ・バッファへのポインタを維持するＬＣＰチャネルを指す。記述子は、ホスト１０のメモリ６０内に配置されることが好ましい。

【００８１】

直接アドレッシングでは、アダプタ８０を介する着信パケットの処理の待ち時間が、実質的に短縮される。しかし、直接アドレッシングでは、アダプタ８０での仮想アドレスから物理アドレスへの変換情報の保管を含めて、ＬＣＰマネージャ１３０によるメモリ・バッファの登録が必要である。ホスト１０のソフトウェアは、チャネル・ドアベルに書き込んで、チャネルによって使用可能な空きバッファに追加されるワード数を示す。直接モードでは、下記のステップを使用して、宛先バッファのアドレスを判定する。

Ａ．アドレスＡが、ＬＣＰエンジンへのコマンドとして駆動される。

Ｂ．（任意選択）アドレスＡをアドレスＡ’にマッピングする。

Ｃ．アドレスＡ’（ステップＢが実行される場合）またはＡ（ステップＢが実行されない場合）が、宛先バッファの基底アドレスになる。

【００８２】

間接モードでは、アダプタ８０が、記述子を使用して、データ・バッファのアドレスを見つける。記述子は、ホスト１０のソフトウェアによって管理される。記述子は、ホスト１０のメモリ６０に配置されることが好ましい。用語「間接」は、アダプタ８０が、宛先アドレスを定義するために追加情報を読み取ることを強調するのに使用される。アダプタ８０は、実行時中にこの情報にアクセスする。間接アドレッシングでは、変換テーブルを保管するのに必要なアダプタ８０のメモリの量が削減される。記述子は、通常は、ホスト１０のメモリ６０に配置される。間接モードでは、宛先バッファのアドレスを判定するのに、下記のステップが使用される。

Ａ．アドレスＡが、ＬＣＰエンジンに対するコマンドとして駆動される。

Ｂ．（任意選択）アドレスＡをアドレスＡ’にマッピングする。

Ｃ．アドレスＡ’（ステップＢが実行される場合）またはＡ（ステップＢが実行されない場合）が、ポインタ記述子のアドレスになる。

Ｄ．バッファへのポインタすなわちアドレスＢを、記述子から読み取る。

Ｅ．（任意選択）アドレスＢをアドレスＢ’にマッピングする。

Ｆ．アドレスＢ’（ステップＥが実行される場合）またはＢ（ステップＥが実行されない場合）が、宛先バッファの基底アドレスになる。

【００８３】

各ＲＸ　ＬＣＰチャネルでは、下記のデータ構造が使用される。

メモリ・マッピングされたアドレスとして実施される、ドアベルへのアクセスによって、追加データについてまたはアダプタ８０がパケット・データを書き込むのに使用可能な記述子について、アダプタ８０が知らされる。

アダプタ８０のメモリ・サブシステム２１０内のＬＣＰコンテキスト項目に、ＬＣＰ属性、状況、構成、および状況フィールドが含まれる。

間接モードで使用されるメモリ・バッファをポイントする記述子。

ホスト１０のメモリ６０内の連続する仮想アドレス空間内のバッファ。

アドレス・マッピング用の、アダプタ８０のメモリ空間２１０内の変換テーブルおよび保護ブロック。

【００８４】

パケットの受信のフローは、下記の特性に依存する。

直接または間接の、アドレッシング・モード。

間接モードについて、記述子がホスト１０のメモリ６０内に配置される。

直接モードについて、アドレス・マッピングを、記述子へのアクセス中に使用してもしなくてもよい。

アドレス・マッピングを、バッファへのアクセス中に使用してもしなくてもよい。

間接モードについて、アドレス保護を、記述子へのアクセス中に使用してもしなくてもよ

い。
アドレス保護を、バッファへのアクセス中に使用してもしなくてもよい。
【００８５】
これらの特性は、ＬＣＰチャネルごとに、ＬＣＰチャネル初期化中にチャネルのコンテキスト１４０の一部としてセットされる。
【００８６】
図１５を参照すると、パケット受信のフローに、ステップ１３００での、受信されたデータ用の空きバッファ１３２０の、ホスト１０内のソフトウェア１３１０による準備が含まれる。ステップ１３３０で、間接モードで、ホスト１０のソフトウェア１３１０が、記述子１３４０を更新する。記述子１３４０は、ホスト１０のメモリ６０に配置される。ステップ１３５０で、ホスト１０のソフトウェアが、ドアベルを鳴らして、空きバッファ空間についてアダプタ８０に通知する。間接モードでは、ドアベルが、新しい記述子１３４０を示す情報を提供する。直接モードでは、ドアベルが、追加された空きバッファ空間を示す情報を提供する。この段階で、アダプタ８０は、ネットワーク３０からホスト１０のメモリ６０に受信データを転送する準備ができている。ステップ１３００、１３３０、および１３５０は、ホスト１０のソフトウェア１３１０が、ＲＸ　ＬＣＰチャネルに空きバッファ１３２０を追加する時に、必ず繰り返される。ＩＳＯＣ１２０は、受信されたパケットごとに下記のステップを繰り返す。ステップ１３６０で、アダプタ８０が、データを受信する。ステップ１３７０で、間接モードで、アダプタ８０が、空きデータ・バッファ１３２０の位置をポイントする記述子１３４０を読み取る。ステップ１３８０で、データおよびヘッダが、データ・バッファ１３２０に書き込まれる。ステップ１３９０で、状況が、完了キュー内で更新される。
【００８７】
図１６を参照すると、受信ＬＣＰチャネル・フローの例で、ポインタ記述子が使用されない。さらに、変換テーブルが使用されない。データ・バッファ１４００で、バッファ１４００を使用するホスト１０内のソフトウェアの物理アドレス空間の連続する空間が使用される。ヘッダおよびデータ・ペイロードの両方が、バッファ１４００に書き込まれる。ドアベルへの書込トランザクション１４１０によって、アダプタ８０による使用のために使用可能なデータ空間が更新される。この情報は、ＬＣＰコンテキスト１４０に保管される。受信／完了ＬＣＰコンテキスト１４０に、新しいデータ／完了項目の書込に使用される次／現在のアドレスへの、バッファ１４００の先頭へのポインタ１４２０とオフセット１４３０が含まれる。アダプタ８０は、パケットを受信する時に、次のパケット位置へのオフセット１４３０を増分し、使用可能なデータ空間を更新する。完了項目１４４０が、フレーム受信の完了時、フレーム・タイムアウト時、またはＬＣＰクライアント１００が知ることを必要とする他のフレーム・イベントについて、完了ＬＣＰ１４５０に追加される。完了項目１４４０には、ＬＣＰデータ・バッファ１４００内でフレームを突き止めるためにＬＣＰクライアント１００が必要とするすべての情報が含まれる。ホスト１０のソフトウェアは、完了項目１４４０内のフィールドを使用して、完了項目１４４０の所有権を与えられていることを認識する。
【００８８】
ＩＳＯＣ１２０は、アダプタ８０のメモリ・サブシステム２１０とホスト１０のメモリ６０の間でデータを移動するのにＬＣＰチャネルを使用できるようにする。ホスト１０のメモリ６０からアダプタ８０にデータを転送するために、送信チャネルが使用される。アダプタ８０からホスト１０のメモリ６０にデータを転送するために、受信チャネルが使用される。データが、ホスト１０のメモリ６０からアダプタ８０に転送される時に、フレーム記述子に、ＩＳＯＣ１２０のバス３９０での宛先アドレスが含まれる。このアドレスによって、フレーム・データ・ペイロードの宛先が定義される。パケット・ヘッダは、通常の形で転送される。これによって、ＩＳＯＣ１２０のメモリ空間へのテーブルおよびコードのロードが可能になる。ＩＳＯＣ１２０のメモリ空間からホスト１０のメモリ６０に受信チャネルを介してデータを転送するために、記述子が、ＲＸプロセッサ１６０によって開

始される。この記述子には、ホスト１０のメモリ６０での宛先アドレスとソース・アドレスの両方を示す情報が含まれる。

【００８９】

上で説明した本発明の好ましい実施形態では、アダプタ８０が、バス・アーキテクチャ７０を介してホスト・コンピュータ・システム１０のＣＰＵ５０およびメモリ６０に接続される。しかし、本発明の他の実施形態では、アダプタ８０を、バス・アーキテクチャ７０とは独立にホスト・コンピュータ・システム１０に統合することができる。たとえば、本発明の他の実施形態で、アダプタ８０を、ホスト・メモリ６０に接続されたメモリ・コントローラを介してホスト・システムに統合することができる。

【００９０】

さらに、上で説明した本発明の好ましい実施形態では、アダプタ８０が、ホスト・コンピュータ・システム１０に挿入されるプラグ可能アダプタ・カードの形で実施された。しかし、本発明の他の実施形態で、アダプタ８０の異なる実施形態が可能であることを諒解されたい。たとえば、アダプタ８０を、ＣＰＵ５０およびメモリ６０と共に、ホスト・コンピュータ・システムのマザー・ボード上に配置することができる。

【図面の簡単な説明】

【００９１】

【図１】データ処理ネットワークの例のブロック図である。

【図２】データ処理ネットワークのネットワーク・インターフェース・アダプタ・カードのブロック図である。

【図３】データ・ネットワークのホスト・システムの例のブロック図である。

【図４】ネットワーク・アダプタ・カードのインテグレーテッド・システム・オン・アチップ（ＩＳＯＣ）の例のブロック図である。

【図５】ＩＳＯＣのもう１つのブロック図である。

【図６】ＩＳＯＣを通る情報のフローを示すＩＳＯＣのブロック図である。

【図７】ＩＳＯＣを通る論理送信パスのブロック図である。

【図８】ＩＳＯＣを通る論理受信パスのブロック図である。

【図９】循環記述子テーブルのブロック図である。

【図１０】記述子テーブルのリンクされた組のブロック図である。

【図１１】仮想バッファおよびその物理的に対応するバッファのブロック図である。

【図１２】完了キューのブロック図である。

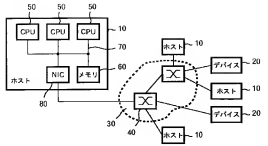
【図１３】ホストからネットワークへのデータの送信フローのブロック図である。

【図１４】ホストからネットワークへのデータの送信フローのもう１つのブロック図である。

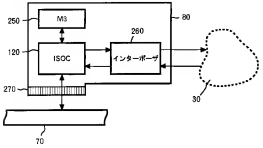
【図１５】ネットワークからホストへのデータの受信フローのブロック図である。

【図１６】ネットワークからホストへのデータの受信フローのもう１つのブロック図である。

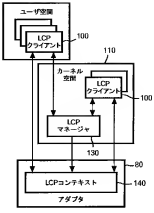
【図１】



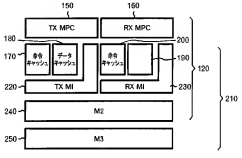
【図２】



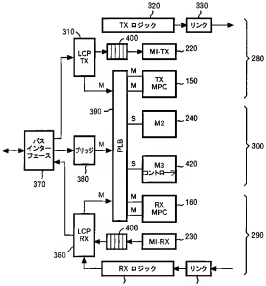
【図３】



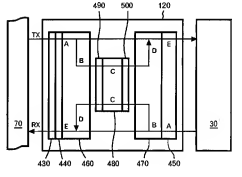
【図４】



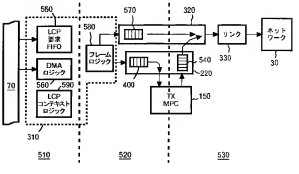
【図５】



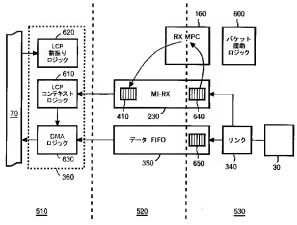
【図６】



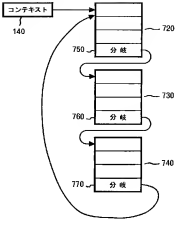
【図７】



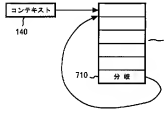
【図８】



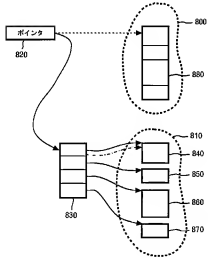
【図１０】



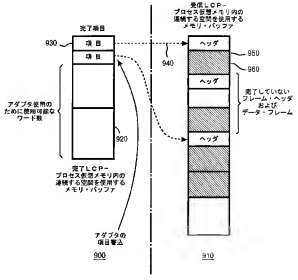
【図９】



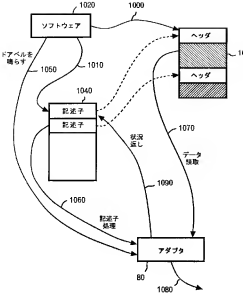
【図１１】



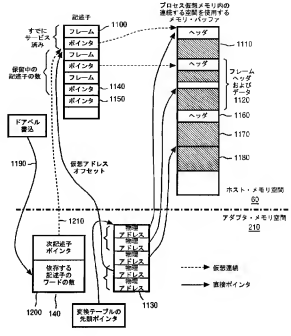
【図１２】



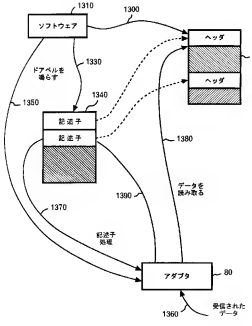
【図１３】



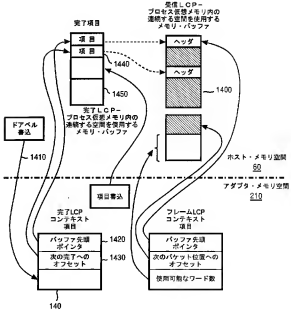
【図１４】



【図１５】



【図１６】



【国際公開パンフレット】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 August 2002 (08.08.2002)

PCT

(10) International Publication Number
WO 02/061592 A1

(51) International Patent Classification[7]: G06F 13/40

(21) International Application Number: PCT/IB01/00122

(22) International Filing Date: 31 January 2001 (31.01.2001)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant (*for all designated States except US*): **INTERNATIONAL BUSINESS MACHINES CORPORATION** [US/US], New Orchard Road, Armonk, NY 10504 (US).

(71) Applicant (*for MC only*): **IBM (SCHWEIZ) AG** [CH/CH]; Baudhweg 21, CH-8010 Zurich (CH).

(72) Inventors; and
(75) Inventors/Applicants (*for US only*): **BIRAN, Giora** [IL/IL]; 13 Inbar Street, Zichron Ya'akov (IL). **SOSTHEIM, Tal** [IL/IL]; Alonim Street 7, 36000 Kiryat Tivon (IL).

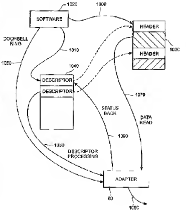
(74) Agent: **WILLIAMS, Jules, David**, International Business Machines Corporation, Saeumerstrasse 4, CH-8803 Rueschlikon (CH).

(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR CONTROLLING FLOW OF DATA BETWEEN DATA PROCESSING SYSTEMS VIA A MEMORY



(57) Abstract: Apparatus is described for controlling flow of data between first and second data processing systems via a memory. The apparatus comprises descriptor logic for generating a plurality of descriptors including a frame descriptor defining a data packet to be communicated between a location in the memory and the second data processing system, and a pointer descriptor identifying the location in the memory. The apparatus also comprises a descriptor table for storing the descriptors generated by the descriptor logic for access by the first and second data processing systems.

WO 02/061592 A1

WO 02/061592 A1

pean (AL, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Published:
— *with international search report*

METHOD AND APPARATUS FOR CONTROLLING FLOW OF DATA BETWEEN DATA PROCESSING SYSTEMS VIA A MEMORY

The present invention relates to a method and apparatus for controlling flow of data via a memory between first and second data processing systems such as a host computer system and a data communications interface for communicating data between the host computer system and a data communications network.

A conventional data processing network comprises a plurality of host computer systems and a plurality of attached devices all interconnected by an intervening network architecture such as an Ethernet architecture. The network architecture typically comprises one or more data communications switches. The host computer systems and the attached devices each form a node in the data processing network. Each host computer system typically comprises a plurality of central processing units and data storage memory device interconnected by a bus architecture such as a PCI bus architecture. A network adapter is also connected to the bus architecture for communicating data between the host computer system and other nodes in the data processing network via the network architecture. It would be desirable for transfer of data and control information between the host computer system and the network architecture to be facilitated as efficiently as possible.

In accordance with the present invention, there is now provided apparatus for controlling flow of data between first and second data processing systems via a memory, the apparatus comprising descriptor logic for generating a plurality of descriptors including a firstas descriptor defining a data packet to be communicated between a location in the memory and the second data processing system, and a pointer descriptor identifying the location in the memory; and a descriptor table for storing the descriptors generated by the descriptor logic for access by the first and second data processing systems.

The descriptor logic and descriptor table improve efficiency of data flow control between the first and second data processing systems such as a host computer system and a data communications interface for communicating data between the host computer system and a data communications network.

The descriptor table may be stored in the memory of the host computer system. Alternatively, the descriptor table is stored in a memory of the data communications interface. The descriptor logic may also generate a branch descriptor comprising a link to another descriptor in the descriptor table. The descriptor table may comprises a plurality of descriptor lists sequentially linked together via branch descriptors therein. Alternatively, the descriptor table may comprise a cyclic descriptor list.

The present invention extends to a data processing system comprising a host processing system having a memory, a data communications interface for communicating data between the host computer system and a data communications network, and apparatus as hereinbefore described for controlling flow of data between the memory of the host computer system and the data communications interface

Viewing the present invention from another aspect, there is now provided a method for controlling flow of data between first and second data processing systems via a memory, the method comprising: by descriptor logic, generating a plurality of descriptors including a frame descriptor defining a data packet to be communicated between a location in the memory and the second data processing system, and a pointer descriptor identifying the location in the memory; and storing the descriptors generated by the descriptor logic in a descriptor table for access by the first and second data processing systems.

Preferred embodiments of the present invention will now be described, by way of example only, with reference to the accompanying drawings , in which:

Figure 1 is a block diagram of an example of a data processing network;

Figure 2 is a block diagram of a network interface adapter card for the data processing network;

Figure 3 is a block diagram of an example of a host computer system for the data network;

Figure 4 is a block diagram of an example of an Integrated System on a Chip (ISOC) for the network adapter card;

Figure 5 is another block diagram of the ISOC;

Figure 6 is a block diagram of the ISOC demonstrating information flow through the ISOC;

Figure 7 is a block diagram of a logical transmit path through the ISOC;

Figure 8 is a block diagram of a logical receive path through the ISOC,

Figure 9A is a block diagram of a cyclic descriptor table

Figure 9B is a block diagram of a linked set of descriptor tables;

Figure 10 is a block diagram of a virtual buffer and its physical counterpart buffer;

Figure 11 is a block diagram of a completion queue;

Figure 12 is a block diagram of a transmit flow of data from the host to the network;

Figure 13 is another block diagram of a transmit flow of data from the host to the network;

Figure 14 is a block diagram of a receive flow of data from the network to the host; and,

Figure 15 is another block diagram of a receive flow of data from the network to the host.

Referring first to Figure 1, an example of a data processing network embodying the present invention comprises a plurality of host computer systems 10 and a plurality of attached devices 20 interconnected by an intervening network architecture 30 such as an InfiniBand network architecture (InfiniBand is a trade mark of the InfiniBand Trade Association). The network architecture 30 typically comprises a plurality of data communications switches 40.

The host computer systems 10 and the attached devices 20 each form a node in the data processing network. Each host computer system 10 comprises a plurality of central processing units (CPUs) 50, and a memory 60 interconnected by a bus architecture 70 such as a PCI bus architecture. A network adapter 80 is also connected to the bus architecture for communicating data between the host computer system 10 and other nodes in the data processing network via the network architecture 30.

Referring now to Figure 2, in particularly preferred embodiments of the present invention, the network adapter 80 comprises a pluggable option card having a connector such as an edge connector for removable insertion into the bus architecture 70 of the host computer system 10. The option card carries an Application Specific Integrated Circuit (ASIC) or Integrated System on a Chip (ISOC) 120 connectible to the bus architecture 70 via the connector 170, one or more third level memory modules 250 connected to the ISOC 120, and an interposer 260 connected to the ISOC 120 for communicating data between the media of the network architecture 30 and the ISOC 120. The interposer 260 provides a physical connection to the network 30. In some embodiments of the present invention, the interposer 260 may be implemented in a single ASIC. However, in other embodiments of the present invention, the interposer 260 may be implemented by multiple components. For example, if the network 30 comprises an optical network, the interposer 260 may comprise a retimer driving a separate optical transceiver. The memory 250 may be implemented by SRAM, SDRAM, or a combination thereof. Other forms of memory may also be employed in the implementation of memory 250. The ISOC 120 includes a first and a second memory. The memory subsystem of the adapter 80 will be described shortly. As will become apparent from the following description, this arrangement provides: improved performance of distributed applications operating on the data processing network; improved system scalability; compatibility with a range of communication protocols; and reduced processing requirements in the host computer system. More specifically, this arrangement permits coexistence of heterogeneous communication protocols between the adapters 80 and the host systems 10. Such protocols can serve various applications, use the same adapter 80, and use a predefined set of data structures thereby enhancing data transfers between the host and the adapter 80. The number of application channels that can be opened in parallel is determined by the amount of memory resources allocated to the adapter 80 and is independent of processing power embedded in the

adapter. It will be appreciated from the following that the ISOC 120 concept of integrating multiple components into a single integrated circuit chip component advantageously minimizes manufacturing costs and in provides reusable system building blocks. However, it will also be appreciated that in other embodiments of the present invention, the elements of the ISOC 120 may be implemented by discrete components.

In the following description, the term Frame refers to data units or messages transferred between software running on the host computer system 10 and the adapter 80. Each Frame comprises a Frame Header and a data payload. The data payload may contain user data, high level protocol header data, acknowledgments, flow control or any combination thereof. The contents of the Frame Header will be described in detail shortly. The adapter 80 processes only the Frame Header. The adapter 80 may fragment Frames into smaller packets which are more efficiently transported on the network architecture 30. However, such fragmentation generally does not transform the data payload.

In particularly preferred embodiment of the present invention, data is transported on the network architecture 30 in atomic units hereinafter referred to as Packets. Each Packet comprises route information followed by hardware header data and payload data. In a typical example of the present invention, a packet size of up to 1024 bytes is employed. Frames of larger size are fragmented into 1024 byte packets. It will be appreciated that in other embodiments of the present invention, different packet sizes may be employed.

In a preferred embodiment of the present invention, communications between the adapter 80 and multiple applications running on the host computer system 10 are effected via a Logical Communication Port architecture (LCP). The adapter 80 comprises a memory hierarchy which allows optimization of access latency to different internal data structures. This memory hierarchy will be described shortly. In preferred embodiments of the present invention, the adapter 80 provides separate paths for outbound (TX) data destined for the network architecture 30 and inbound (RX) data destined for the host computer system 10. Each path includes it own data transfer engine, header processing logic and network architecture interface. These paths will also be described in detail shortly.

Referring now to Figure 3, the LCP architecture defines a framework for the interface between local consumers running on the host computer system 10 and the adapter 80. Examples of such consumers include both applications and threads. The computer system 10 can be subdivided into a user application space 90 and a kernel space 110. The LCP architecture provides each consumer with a logical port into the network architecture 80. This port can be accessed directly from a user space 90. In particularly preferred embodiments of the present invention, a hardware protection mechanism takes care of access permission. An LCP registration is performed by in the kernel space 110 prior to transfer of data frames. The LCP architecture need not define a communication protocol. Rather, it defines an interface between the applications and the adapter 80 for transfer of data and control information. Communication protocol details may be instead set by the application and program code executing in the adapter 80. The number of channels that can be used on the adapter 80 is limited only by the amount of memory on the adapter card 80 available for LCP related information. Each LCP port can be programmable to have a specific set of features. The set of features is selected according to the specific protocol to best support data transfer between the memory 60 in the host computer system and the adapter 80. Various communication protocols can be supported simultaneously, with each protocol using a different LCP port.

The LCP architecture comprises LCP Clients 100, an LCP Manager 130 resident in the kernel space 130, and one or more LCP Contexts 140 resident in the adapter 80.

Each LCP Client 100 is a unidirectional application end point connected to an LCP port. An LCP client 100 can be located in the user application space 90 or in the kernel 110. In operation, each LCP client 100 produces commands and data to be read from the memory 60 and transferred by the adapter 80 via a TX LCP channel, or consumes data transferred by the adapter 80 to the memory 60 via an RX LCP channel.

The LCP Manager 130 is a trusted component that services request for LCP channel allocations and deallocations and for registration of read/write areas in the memory 60 for each channel. The LCP Manager 130 allows a user space application to use resources of the adapter 80 without compromising other communication operations, applications, or the operating system of the host computer system 10.

Each LCP Context 140 is the set of control information required by the adapter 80 to service a specific LCP Client 100. The LCP Context 140 may include LCP channel attributes which are constant throughout existence of the channel, such as possible commands, pointer structure, and buffer descriptor definitions. The LCP Context 140 may also include specific LCP service information for the LCP channel, such as the amount of data waiting for service, and the next address to access for the related LCP channel. The LCP context 140 is stored in memory resident in the adapter 80 to enable fast LCP context switching when the adapter 80 stops servicing one channel and starts servicing another channel.

An LCP Client 100 requiring initiation of an LCP port turns to the LCP Manager 130 and requests the allocation of an LCP channel. The LCP channel attributes are determined at this time and prescribe the behavior of the LCP port and the operations that the LCP Client 100 is authorized to perform in association with the LCP port. The LCP Client 100 is granted an address that will be used to access the adapter 80 in a unique and secure way. This address is known as a Doorbell Address.

The LCP Manager 130 is also responsible for registering areas of the host memory 60 to enable virtual to physical address translation by the adapter, and to allow user space clients to access these host memory areas without tampering with other programs.

Registration of new buffers and deregistration of previous buffers can be requested by each LCP Client 100 during run-time. Such a change, requires a sequence of information exchanges between the LCP Client 100, the LCP Manager 130, and the adapter 80.

Each LCP Client 100 and port are associated with an LCP Context 140 that provides all the information required by the adapter 80 to service pending requests sent by the LCP port for command execution.

To initiate memory transfers between the LCP Client 100 and the adapter 80, and initiate transmission of frames, the LCP Client 100 prepares descriptors holding the information for a specific operation. The LCP Client 100 then performs an I/O write to the Doorbell address

mapped to the adapter 80. Writing to the Doorbell address updates the LCP Context 140 on the adapter 80, adding the new request for execution.

The adapter 80 arbitrates between various transmit LCP pools that have pending requests, and selects the next one to be serviced.

On receipt of data, the Frame and LCP for a received packet are identified. Descriptors are generated to define the operation required for the receive LCP. Execution of these descriptors by an LCP Engine of the adapter 80 stores the incoming data in an appropriate data buffer allocated to the LCP channel in the memory 60 of the host computer system 10.

For each LCP channel serviced, the adapter 80 loads the associated LCP context information and uses this information to perform the desired set of data transfers. The adapter 80 then continues on to process the next selected LCP Context 140.

Referring now to Figure 3, and as mentioned earlier, the ISOC 120 comprises a first memory space 220 and 230 and a second memory space 240 and the adapter 80 further comprises a third level memory 250. The first, second, and third memory spaces for part of a memory subsystem 210 of the adapter 80. In a preferred embodiment of the present invention, the ISOC 120 comprises a TX processor (TX MPC) 150 dedicated to data transmission operations and an RX processor (RX MPC) 160 dedicated to data reception operation. In particularly preferred embodiments of the present invention, processors 150 and 160 are implemented by Reduced Instruction Set Computing (RISC) microprocessors such as IBM PowerPC 405 RISC microprocessors. Within the memory subsystem 210, the ISOC 120 comprises, in addition to the first and second memory spaces, a data cache 180 and an instruction cache 170 associated with TX processor 150, together with a second data cache 190 and second instruction cache 190 associated with RX processor 160. The difference between the three levels is the size of memory and the associated access time. As will become apparent shortly, the memory subsystem 210 facilitates: convenient access to instruction and data by both the TX processor 150 and the RX processor 160; scaleability; and sharing of resources between the TX processor 150 and the RX processor 160 in the interests of reducing manufacturing costs.

The first level memory spaces (M1) 220 and 230 comprise a TX-M1 memory space 220 and RX-M1 memory space 230. The TX-M1 memory 220 can be accessed only by the TX processor 150 and the RX-M1 memory 230 can be accessed only by the RX processor 160. In operation the first level memory spaces 220 and 230 are used to hold temporary data structures, header templates, stacks, etc. The first level memory spaces 220 and 230 both react to zero wait states. Each one of the first level memory spaces 220 and 230 is connected only to the data interface of the corresponding one of the processors 150 and 160 and not to the instruction interface. This arrangement enables both cacheable and non-cacheable first level memory areas available while maintaining efficient access to data in the first level memory spaces 230 and 240.

The second level memory space (M2) 240 is a shared memory available to both processors 150 and 160, other components of the adapter 80, and to the host computer system 10. Access to the second level memory space 240 is slower than access to the first level memory areas 220 and 230 because the second level memory space 240 is used by more agent via a shared internal bus. The third level memory space 250 is also a shared resource. In particularly preferred embodiments of the present invention the adapter 80 comprises a computer peripheral circuit card on which the first level memory spaces 220 and 230 and the second level memory space 240 are both integrated on the same ASIC as the processors 150 and 160. The shared memory spaces 240 and 250 are generally used for data types that do not require fast and frequent access cycles. Such data types include LCP contexts 140 and virtual address translation tables. The shared memory spaces 240 and 250 are accessible to both instruction and data interfaces of the processors 150 and 160.

The adapter 80 handles transmission and reception data flows separately. The seperate processor 150 and 160 for the transmission and reception path avoids the overhead of switching between task, isolates temporary processing loads in one path from the other path, and facilitates use of two embedded processors to process incoming and outgoing data streams. Referring now to Figure 5, the ISOC 120 comprises transmission path logic 280 and reception path logic 290, and shared logic 300. The transmission path logic 280 comprises an LCP TX engine 310 for decoding specifics of each LCP channel and fetching LCP related

commands for execution; TX logic 320 for controlling transfer of frames into the adapter 80, the aforementioned TX processor 150 for managing TX frame and packet processing; the aforementioned first level TX memory 220 for holding instructions and temporary data structures; and link logic 330; and logic for assisting the TX processor 150 in managing the data flow and packet processing such as routing processing for fragmentation of frames into data packets. The TX processor 150 processes tasks in series based on a polling only scheme in which the processor is interrupted only on exceptions and errors. The first level TX memory 220 is employed by the processor 150 for communicating with TX logic 320. The reception path logic 290 comprises link logic 360; hardware for assisting the aforementioned RX processor 160 in processing headers of incoming packets and transformation or assembly of such packets into frames; the aforementioned RX processor 160 for RX frame and packet processing; the aforementioned first level RX memory 230 for holding instructions: RX logic 350 for controlling transfer of frames from the network architecture 30; and an LCP RX engine 360 for decoding the specifics of each LCP channel, storing the incoming data in the related LCP data structures in the memory 60 of the host computer system and accepting and registering pointers to empty frame buffers as they are provided by the LCP Client 100 for use by the adapter 80. The RX processor 160 processes tasks in series using a polling only scheme in which the RX processor 160 is interrupted only on exceptions or errors. The level 1 RX memory 230 is used by the RX processor 160 to communicate with the RX logic 350.

As mentioned earlier, the ISOC approach permits reduction in manufacturing costs associated with the adapter 80 and the other components thereof, such as the circuit board and the other supporting modules. The ISOC approach also increases simplicity of the adapter 80, thereby increasing reliability. The number of connections between elements of the ISOC 120 is effectively unlimited. Therefore, multiple and wide interconnect paths can be implemented. In the interests of reducing data processing overheads in the host computer system 10, data transfer operations to and from the host memory 60 are predominantly performed by the ISOC 120. The ISOC 120 also performs processing of the header of incoming and outgoing packets. During transmission, the ISOC 120 builds the header and routes it to the network architecture 30. During reception, the adapter 80 processes the header in order to determine its location in the system's memory. The level 1 memories 220 and 230 are zero wait state memories providing processor data space such as stack, templates, tables, and temporary storage

locations. In especially preferred embodiments of the present invention, the transmission path logic 280, reception path logic 290, and shared logic 300 are built from smaller logic elements referred to as cores. The term core is used because these elements are designed as individual pieces of logic which have stand-alone properties enabling them to be used for different applications.

As indicated earlier, the transmission path logic 280 is responsible for processing transmission or outgoing frames. Frame transmission is initiated via the bus architecture 70 by a CPU such as CPU 50 of the host computer system 10. The ISOC 120 comprises bus interface logic 370 for communicating with the bus architecture 70. The ISOC 120 also comprises bus bridging logic 390 connecting the bus interface logic 370 to a processor local bus (PLB) 390 of the ISOC 120. The TX LCP engine 310 fetches commands and frames from the host memory 60. The TX processor 150 processes the header of each frame into a format suitable for transmission as packets on the network architecture 30. The TX logic 320 transfer the frame data without modification. The link logic 330 processes each packet to be transmitted into a final form for transmission on the network architecture 30. The link logic 330 may comprises one or more ports each connectable to the network architecture 30.

As indicated earlier, the reception path logic 290 is responsible for processing incoming packets. Initially, packets received from the network architecture 30 are processed by link logic 340. Link logic 340 recreates the packet in a header and payload format. To determine the packet format and its destination in the host memory 60, the header is processing by the RX processor 230. The link logic 340 may comprises one or more ports each connectable to the network architecture 30. The RX LCP engine is responsible for transferring the data into the host memory 60 via the bus architecture 70.

The transmission path logic 280 comprises a HeaderIn first in- first out memory (FIFO) 400 between the TX LCP engine 310 and the TX processor 220. The reception path logic comprises a HeaderOut FIFO 410 between the RX processor 230 and the RX LCP engine 360. Additional FIFOs and queues are provided in the TX logic 320 and the RX logic 350. These FIFOs and queues will be described shortly.

The shared logic 300 comprises all logical elements shared by the transmission path logic 280 and the reception path logic 290. These elements include the aforementioned bus interface logic 370, bus bridging logic 380, PLB 390, second level memory 240 and a controller 420 for providing access to the remote third level memory 250. The bus interface logic 370 operates as both master and slave on the bus architecture 70. As a slave, the bus interface logic allows the CPU 50 to access the second level memory 240, the third level memory 250 via the controller 420, and also configuration registers and status registers of the ISOC 120. Such registers can generally be accessed by the CPU 50, the TX processor 150 and the RX processor 160. As a master, the bus interface logic allows the TX LCP engine 310 and the RX LCP engine 360 to access the memory 60 of the host computer system 10. In Figure 5, "M" denotes a master connection and "S" denotes a slave connection.

Referring now to Figure 6, packet flow through the ISOC 120 is generally symmetrical. In other words, the general structure of flow is similar in both transmit and receive directions. The ISOC 120 can be regarded as comprising first interface logic 440; a first control logic 460; processor logic 480; second control logic 470; and second interface logic 450. Packets are processed in the following manner:

A. In the transmit direction, information is brought into the ISOC 120 from the bus architecture 70 through the first interface logic. In the receive direction, information is brought into the ISOC 120 from the network architecture 30 through the second interface logic 450.

B. In the transmit direction, information brought into the ISOC 120 through the first interface logic 440 is processed by the first control logic 460. In the receive direction, information brought into the ISOC through the second interface logic 450 is processed by the second control logic 470.

C. In the transmit direction, a frame header is extracted for an outgoing frame at the first control logic 460 and processed by the processor logic 480. The processor logic 480 generates instructions for the second control logic 470 based on the frame header. The payload of the outgoing frame is passed to the second interface logic 470. In the

receive direction, a frame header is extracted from an incoming frame at the second control logic 470 and processed by the processor logic 480. The processor logic 480 generates instructions for the first control logic 460 based on the frame header. The payload of the incoming frame is passed to the first control logic 460. In both directions, the processor 480 is not directly handling payload data.

D. In the transmit direction, the second control logic 470 packages the outgoing payload data according to the instructions received from the processor logic 480. In the receive direction, the first control logic 460 packages the incoming payload according to the instructions received from the processor logic 480.

E. In the transmit direction, the information is moved through the second interface logic 450 to its destination via the network architecture 50. In the receive direction, the information is moved through the first interface logic to its destination via the bus architecture 70.

An interface to software operating on the host computer system 10 is shown at 430. Similarly, interfaces to microcode operating on the processor inputs and outputs is shown at 490 and 500.

Referring to Figure 7, what follows now is a more detailed description of one example of a flow of transmit data frames through the ISOC 120. The ISOC 120 can be divided into an LCP context domain 510, a frame domain 520 and a network domain 530 based on the various formats of information within the ISOC 120. The TX LCP engine 310 comprises an LCP requests FIFO 550, Direct Memory Access (DMA) logic 560, frame logic 580, and the aforementioned LCP context logic 140. The LCP request FIFO 550, DMA logic 560, and LCP TX Context logic 590 reside in the LCP context domain 510. The frame logic 580 resides in the frame domain 520. The TX logic 320, first level TX memory space 220, and TX processor 150 straddle the boundary between the frame domain 520 and the network domain 530. The TX link logic 330 resides in the network domain 530. In particularly preferred embodiments of the present invention, the Henderin FIFO 400 is integral to the first level TX memory space 220. In general, an application executing on the host computer system 10 creates a frame. The

frame is then transmitted using a TX LCP channel on the adapter 80. Handshaking between the application and the adapter 80 assumes a prior initialization performed by the LCP Manager 130. To add an LCP Service Request, an LCP Client 100 informs the adapter 80 that one or more additional transmit frames are ready to be executed. This is performed by writing to a control word in to a Doorbell. The Doorbell's addresses are allocated in such as way that the write operation is translated into a physical write cycle on the bus architecture 70, using an address that is uniquely associated with the LCP port and protected from access by other processes. The adapter 80 detects the write operation and logs the new request by incrementing an entry of previous requests for the specific LCP Client 100. This is part of the related LCP Context 140. An arbitration list, retained in the memory subsystem 210 of the adapter 80 is also updated. In a simple example, arbitration uses the aforementioned FIFO scheme 550 between all transmit LCP channels having pending requests. While one LCP channel is serviced, the next LCP channel is selected. The service cycle begins when the corresponding LCP Context is loaded into the TX LCP engine 310. The LCP Context 140 is then accessed to derive atomic operations for servicing the LCP channel and to determine parameters for such operations. For example, such atomic operations may be based on LCP channel attributes recorded in the LCP Context 140. A complete service cycle typically includes a set of activities performed by the adapter 80 to fetch and execute a plurality of atomic descriptors created by the LCP Client 100. In the case of a TX LCP channel, the service cycle generally includes reading multiple frames from the host memory 60 into the memory subsystem 210 of the adapter 80. Upon conclusion, all the LCP Context information requiring modification (in other words, the LCP Service Information) is updated in the memory subsystem 210 of the adapter 80. In general, the first action performed by the adapter 80 within the LCP Service cycle, is to fetch the next descriptor to be processed.

Processing of transmission frames by the ISOC 120 typically includes the following steps:

A. Fetching the subsequent LCP port frame descriptor.

The address of the next descriptor to be fetched is stored as parts of the LCP channel's Context 140. The adapter 80 reads the descriptor from host memory 60 and decodes

the descriptor based on the LCP channel attributes. The descriptor defines the size of the new frame header, the size of the data payload, and the location of these items.

B. Conversion of virtual address to physical address.

If a data buffer is referenced by virtual memory addresses in an application, the address should go through an additional process of address translation. In this case, the virtual address used by the application is translated into a physical address usable by the adapter 80 while it access the host memory 60. This is done by monitoring page boundary crossings and using physical page location information written by the LCP manager 130 into the memory subsystem 210 of the adapter 80. The virtual to physical translation process serves also as a security measure in cases where a descriptor table is created by an LCP client 100 which is not trusted. This prevents unauthorized access to unrelated areas of the host memory 60.

C. Reading the frame header.

Using physical addressing, the header and payload data of the TX frame are read from buffers in the host memory 60. The header is then stored in the TX Headerln FIFO 400. When the header fetch is completed, the adapter 80 sets an internal flag indicating that processing of the header can be initiated by the TX processor 150.

D. Reading the frame data.

The payload data is read from the host memory 60 and stored by the adapter 80 in a data FIFO 570. The data FIFO 570 is shown in Figure 7 as resident in the TX logic 320. However, the data FIFO 570 may also be integral to the first level TX memory space 220. Data read transactions continue until all data to be transmitted is stored in the memory subsystem 210 of the adapter 80. Following completion of the read operation, a status indication is returned to the LCP Client 100. Note that processing of the header can start as soon as the header has been read into the Headerln FIFO 400. There is no need to wait for the whole data to be read.

E. Processing the frame header

The header processing is performed by the TX processor 150. Header processing is protocol dependent and involves protocol information external to the ICP architecture. The TX processor 150 runs TX protocol header microcode and accesses routing tables and other relevant information already stored in the memory subsystem 210 of the adapter 80 during a protocol and routing initialization sequence. When the TX processor 150 receives an indication that a new header is waiting in the HeaderIn FIFO 400, it starts the header processing. The header processing produces one or more packet headers which are in the format employed to send packets over the network architecture 50 and include routing information. If the payload size is larger than a maximum packet size allowed by the network architecture 50, the payload is fragmented by generating several packet headers each used in connection with consecutive data segments of the original payload data to form packets for communication over the network architecture 50.

F. Queuing the packet header for transmission

A command defining the number of header words and the number of data words for a packet and the packet header itself are written by the TX processor 150 to a TX HeaderOut FIFO 540 in the first level memory space 220.

G. Merging packet header and packet data for transmission.

Transmission of a packet on the network architecture 50 is triggered whenever a command is ready in the HeaderOut FIFO 540, and the data FIFO 570 contains enough data to complete the transmission of the related packet. A Cyclic Redundancy Check (CRC) may be added to the header and data of each packet. Each complete packet is transferred to the network architecture 50 via the TX link logic 530.

The transmission process for each frame is completed when all the frame data is transmitted on the network architecture 50, by means of one or more packets. For each frame processed by

the adapter 80, a status may be returned to the application via a second LCP Client 100. This status indicates the completion of the frame data transfer from the host memory 60 onto the adapter 80, completion of the frame transmission itself, or other levels of transmission status.

At any instance in time, the adapter 80 may be concurrently executing some or all of the following actions: selecting the next LCP to be serviced; initiating service for LCP channel A; executing DMA fetch of data for the last frame of LCP channel B; processing a frame header and fragmentation for LCP channel C ; and, transmitting packets originated by LCP channel D.

Referring to Figure 8, what follows now, by way of example only, is a description of a data frame reception by an application using an RX LCP port. The operation of the ISOC 120 may vary depending on the type of protocol supported by the LCP. Handshaking between the application and the adapter 80 assumes a prior initialization performed by the LCP manager 130. The RX LCP engine 360 comprises LCP allocation logic 620, LCP Context logic 610, and DMA logic 630 all residing in the LCP domain 520. The RX processor 160, first level RX memory space 230, and RX logic 350 all straddle the boundary between the frame domain 520 and the network domain 530. The RX link logic 340 and packet assist logic 600 reside in the network domain 530. In particularly preferred embodiments of the present invention, the HeaderOut FIFO 410 is located in the first level RX memory space 230. Frames received by the ISOC 120 from the network architecture 30 are written into LCP client buffers in the host memory 60. Availability of memory buffers is determined by the LCP RX client 100 and is indicated to the adapter 80 for insertion of incoming data frames. The LCP client 100 provides buffers by writing into a receive Doorbell on the ISOC 120, similar to the aforementioned manner in which the transmission path logic 280 is informed of new frames ready to be transmitted. The Doorbell register address is allocated such that the write operation is translated into a physical write cycle on the bus architecture 70. The adapter 80 detects the write operation and logs the new provision of empty memory areas by incrementing the number of available word entries for the specific LCP RX Client 100. The available word count is part of the related LCP context 140. Whenever an application completes processing of a received frame within a buffer, it writes to the Doorbell. The write cycle indicates the number of words in the newly available memory space. The count within the LCP context is

incremented by that amount. A packet received from the network architecture 30 may be part of a larger frame that will be assembled by the adapter 80 into contiguous space in the host memory 60. Processing of received frames by the ISOC 120 generally includes the following steps:

A. Splitting packet header and data

The RX link logic 540 translates information from the network architecture 30 into a stream of packets. Each received packet is processed by the RX link logic 540 to separate the packet header from the payload data. The header is pushed into an RX Headerin FIFO 640 in the first level RX memory space 230. The payload is pushed into an RX data FIFO 650 in the RX logic 350. The RX data FIFO 650 may also be implemented in the first level RX memory space 230.

B. Decoding the packet header and generating and LCP frame header.

The packet header is decoded to provide fields indicative of an ID for the frame to which the packet belongs, the size of the payload, and the size of the frame data. Once the packet header is reader for the RX Headerin FIFO 640, an indication is sent to the RX processor 160. The RX processor processes the packet header information and generates an LCP related command including information required to transfer the packet data. Such information includes packet address and length. At the end of the header processing, a descriptor, or a set of descriptors, are written to the LCP RX HeaderOut FIFO 410, and an indication is triggered.

C. Transfer of data within the RX LCP Context.

The descriptors are fetched from the RX HeaderOut FIFO 410 by the RX LCP engine 360, and then decoded. The descriptors include the LCP number, packet address, packet data length and the source address of the data to be transferred in the memory subsystem 210 of the adapter 80. The RX LCP engine 540 uses the LCP Context

information to create a target physical address (or addresses if a page is crossed) to be written to in the host memory 60 and initiates DMA transfers to write the data.

D. ISOC DMA transactions.

The ISOC 120 aims to optimize transactions on the bus architecture 70 by selecting appropriate bus commands and performing longest possible bursts.

At any instance in time, the adapter 80 may be concurrently executing some or all of the following: processing a buffer allocation for LCP channel X; initiating an inbound data write service for LCP channel A; executing a DMA store of data for LCP channel B; processing a frame assembly of a packet destined for LCP channel C; and, receiving packets for LCP channel D.

To minimize frame processing overhead on the RX processor 160 and TX processor 150, packet assist logic 600 comprises frame fragmentation logic, CRC and checksum calculation logic, and multicast processing logic.

The data flow between both the TX and RX LCP engines 310 and 360 and the host 10 will now be described in detail. Both TX and RX LCP ports use memory buffers for transferring data and descriptor structures that point to such memory buffers. The descriptor structures are used to administer data buffers between a data provider and a data consumer and to return empty memory buffers to be used by the data provider. The descriptors point to the memory buffers based on either physical or virtual addresses.

TX LCP channels are responsible for data transfer from the host memory 60 into buffers of the ISOC 120. Other layers of logic are responsible for transferring data from buffers of the ISOC 120 into the network 30. RX LCP channels are responsible for transferring data received from the network 30 to the host memory 60.

The TX and RX LCP engines 310 and 360 are capable off handling a relatively large number of LCP channels. Each LCP channel has a set of parameters containing all information

specific thereto. The information comprises the configuration of the channel, current state and status. The LCP context 140 associated with a channel is set by the LCP manager 130 during initialization of the channel. During channel operation, the content of the LCP context 140 is updated only by the ISOC 120. The LCP contexts 140 are saved in a context table within the memory subsystem 210 of the adapter 80. Access to the LCP context 140 of an LCP channel is performed according to the LCP number. The LCP RX and TX channels use different LCP context structures.

Data buffers are pinned areas in the memory 60 of the host 10. Transmit buffers hold data that for transmission. The TX LCP engine 310 moves the data located in these buffers into internal buffers of the ISOC 120. Incoming data received from the network 30 is moved by the RX LCP engine 360 into buffers in the memory 60 of the host 10. Ownership of the buffers alternates between software in the host 10 and the ISOC 120. The order of events on LCP TX channels is as follows:

A. Software in the host 10 prepares buffers with data to be transmitted in the memory 60 of the host 10;
B. The software notifies the ISOC 120 that data in the buffers is ready to be transmitted;
C. The ISOC 120 reads the data from the buffers; and,
D. The ISOC 120 identifies to the software in the host 10 the buffers that were read and can be reused by the software in the host 10 to transfer new data.

The order of events on LCP RX channels is as follows:

A. The software in the host 10 prepares buffers into which the ISOC 210 can write the received data;
B. The software notifies the ISOC 120 that free buffers are ready in the memory 60 of the host;
C. The ISOC 120 writes the data to the buffers; and,
D. The ISOC 120 identifies to the software in the host 10 the buffers that were filled with received data and can be processed by the software.

When the software prepares buffers to be used by the ISOC 120, buffer information is tracked via doorbell registers. Information relating to buffers used by the ISOC 120 is returned to the software using a status update or through a completion queue. For TX LCP channels, the buffers include data and header information transferred by the TX LCP engine 310 into the ISOC 120 and processed to become one or more packets for transmission on the network 30. The header is used by the TX processor 150 of the ISOC 120 to generate the header of the packet to be transmitted on the network 30. For RX LCP channels, free buffers are assigned by the software in the host 10 to the adapter 80. The adapter 80 fills the buffers with the received packets.

The descriptors have defined data structures known to both the ISOC 120 and software in the host 10. The software uses descriptors to transfer control information to the ISOC 120. The control information may be in the form of a frame descriptor, a pointer descriptor , or a branch descriptor depending on desired function. Descriptor logic in the software and in the ISOC 120 generate and modify the descriptors according to control measures to be taken. Such measures will be described shortly. A frame descriptor comprises a description of the packet (e.g., data length, header length, etc.). A pointer descriptor comprises a description of a data location. A branch descriptor comprises description of the descriptor location (e.g., link lists of descriptors). Information in the descriptors is used for control by the software in the host 10 of the data movement operations performed by the TX and RX LCP engines 310 and 360. The information used to process a frame to generate a TX packet header is located in the header of the frame. Referring to Figure 9A, descriptors may be provided in a single table 700 with the LCP context 140 pointing to the head of the table 700. Referring to Figure 9B, descriptors may also be arranged in a structure of linked descriptor tables 720-740. Following LCP channel initialization, the LCP context 140 points to the head of the first descriptor table 720 in the structure. Branch descriptors 750-770 are used to generate a linked list of tables 720-740 where a branch descriptor 750-770 at the end of a descriptor table 720-740 points to the beginning of another table 720-0740. Referring back to Figure 9A, branch descriptors can also be used to generate a cyclic buffer where a branch descriptor 710 at the end of a table 700 points to the beginning of the same table 700. A cyclic buffer may also be used in the receive path. In this case, the LCP 140 context is initiated to point to the head of the buffer. The buffer is wrapped around when the ISOC 120 reaches its end. The software in the host 10 can

write the descriptors into the memory 60 in the host 10 (for both the receive and the transmit paths) or into the memory 250 of the adapter 80 (for the transmit path only). Writing descriptors to the memory subsystem 210 of the adapter 80 involves an I/O operation by the software in the host 10 and occupies the memory subsystem 210 of the adapter 80. Writing descriptors in the memory 60 of the host 80 requires the adapter 80 to access the memory 60 of the host 10 whenever it has to read a new descriptor. The location of the software descriptors is defined by the LCP manager 130 for each LCP channel independently. The location of the descriptors is defined according to system performance optimization. The descriptors provide flexibility in the construction of queues.

The RX and TX LCP engines 310 and 360 use addresses to access the descriptors in the descriptor tables and to access data buffers. An address can be either a physical address or a virtual address. The term physical address describes an address that the ISOC 120 can drive, as is, to the bus 70. The term virtual address describes an address which is not a physical one and is used by the software or microcode. The virtual address has to pass through a mapping in order to generate the physical address. An address used by the TX and RX LCP engines 310 and 360 can have different sources as follows: pointer in the LCP channel context 140; pointer in descriptors prepared by software running on the host 10; pointer in descriptors prepared by the RX processor 160; and, pointer in descriptors prepared by the TX processor 150 (used for returning a completion message). A pointer can point to a descriptor or to a data buffer. Every address used by the TX and RX LCP engines 310 and 360 can be optionally mapped to a new address used as the physical address on the bus 70. The address mapping is done by the TX and RX LCP engines 310 and 360. The ISOC 120 uses local memory 210 to hold the translation tables. The LCP manager 130 writes the translation tables to the adapter 80 during memory registration. The address mapping allows virtual addressing to be used for buffers or descriptor tables. The virtual addressing enables the management of virtual buffers that are physically located in more than one physical page. The address mapping also allows the host 10 to work directly with applications using virtual addresses without requiring a translation processor for the software.

Referring to Figure 10, shown therein is an image 800 of a buffer 880 as it appears to the software in the host 10. Also shown is a physical mapping 810 of the address as it is used to

access the memory 60 in the host 10. A virtual pointer points 830 to a location in the buffer. The buffer in this example is a virtual buffer occupying a few noncontiguous pages 840-870 in the memory 60 of the host 10. The LCP engines 310 and 360 perform the mapping by translating the address via a translation table 830. The translation table holds a physical address pointer to the head of each physical buffer 840-870 mapped from the virtual buffer 830. Address mapping in the adapter 80 allows flexibility when mapping descriptors and data buffers in the memory 60 in the host 10. Address mapping in the adapter 80 also allows a direct connection to software buffers that use virtual addresses without requiring the software in the host 10 to perform address translation to a physical address.

Each packet which the adapter 80 writes to the memory 60 in the host has a status associated therewith. The status allows synchronization between the adapter 80 and the software in the host 10. The status can be used to indicate different reliability levels of packets. The ISOC 120 provides the following status write backs. Transmit DMA Completion indicates that a data in a TX packet has been read into the adapter 80; Reliable Transmission is returned to indicate the completion of data transmission in the network 30; Receive DMA Completion indicates completion of a receive data transfer into the memory 60; and, Reliable Reception indicates reception of a transmit packet by a destination node in the network 30.

A TX frame descriptor includes a 2 byte status field. Status write back means that a transaction status is written back into a descriptor. The status includes a completion bit which can be polled by the software in the host 10. When the software in the host 10 finds a set completion bit, it may reuse the buffers associated with the frame defined by the frame descriptor.

A completion queue is implemented by an RX LCP channel. The LCP channel used by the completion queue has all the flexibility and properties that can be implemented by any RX LCP channel. The TX and RX processor 150 and 160 generates status write backs to indicate reliable transmission, reliable reception, receive DMA completion, or transmit DMA completion. Different indications relating to the frame are used in different cases. For example, in the case of a reliable transmission, the TX processor 150. Reads internal registers indicating the status of a packet transmission. In the case of reliable reception, the RX

processor 160 gets a completion indication as a received packet which includes an acknowledgment. In the case of a receive DMA completion, the RX processor 160 uses frame completion information. In the case of a transmit DMA completion, the TX processor 160 indicates the reception of a frame for transmission in the adapter 80. A completion queue can be used by a single TX or RX LCP channel or may shared by multiple channels. Micro code in the adapter 80 updates a status queue by initiating a frame descriptor into a command queue of the RX LCP engine 360. Referring to Figure 11, the status is transferred to the memory 60 of the host 10 via a completion status LCP 900 comprising a completion queue 920. The completion queue 900 is continuous (either physically or virtually) and is located in the memory 60 of the host 10. For example, the completion queue can be held in a continuous buffer. Entries 930 in the completion queue preferably have a fixed size. Each entry holds a pointer 940 to the head of a buffer 950 associated with a receive LCP 910. The buffer 930 is filled by the packet 950 associated with the completion status. .

A TX software/adapter handshake comprises an TX LCP port and an completion RX LCP port. Each LCP transmit channel uses the following data structures:

A Doorbell entry, implemented as a memory mapped address, informs the adapter 80 of incremental requests to process descriptors and data. Each process has a unique access into a single page of memory mapped address used for Doorbell access.

An LCP context entry in the adapter memory space 210 , containing LCP attributes and status fields.

A structure of transmit descriptors. This structure may span across multiple physical pages in the memory 60 of the host 10. If virtual addressing is used for the descriptors, a translation table is used to move one page to the next. If physical addressing is used for the descriptors, branch descriptors are used to move from one page to the next. Transmit descriptors contain a status field that can be updated following transfer of all descriptor related data to the adapter 80.

Transmit data buffers pinned in the memory 60 of the host 10 pointed to by the pointer descriptors. If virtual addressing is used for the data buffers, a translation table converts the pointer into physical addresses used by the adapter 80 to access the memory 60 in the host 10.

A translation table and protection blocks in the adapter memory space 210 are used for address mapping.

Referring to Figure 12, a transmit packet flow comprises, at step 1000, software 1020 in the host 10 filling buffer 1030 with data to be transmitted. At step 1010, the software 1020 updates the descriptors 1040. The descriptors 1040 may be either in the memory 60 of the host 10 or in the memory subsystem 210 of the adapter 80. At step 1050, the software 1020 rings the Doorbell to notify the adapter 80 that new data is ready to be transmitted. At step 1060, the adapter 80 manages arbitration between requests from the different LCP channels. When a channel wins the arbitration, the adapter 80 reads the new descriptors 1040. At step 1070, the adapter 80 reads the data. At step 1080, the data is transmitted to the network 30. At step 1090, the status is updated in the descriptors 1040 or in the completion queue

The TX LCP channel may use address translation when accessing data buffers. In this case, the data buffer is composed of multiple memory pages. As far as the process is concerned, these memory pages are in consecutive virtual memory space. However, as far as the adapter 80 is concerned, these memory pages may be in nonconsecutive physical memory space. A completion status structure contains information indicative of the status of transmitted frames. This is implemented as a separate LCP channel. The frame descriptor, which is the first descriptor for every frame, has an optional status field which can be updated after the frame has been transferred to the adapter 80.

Referring now to Figure 13, in an example of a transmit LCP channel flow, descriptors 1100 are located in the memory 60 of the host 10. Access to the descriptors 1110 and buffers 1110 storing packets 1120 requires address translation through a translation table 1130 located in the adapter 80. The buffers 1110 use contiguous space in the virtual address space of the software in the host 10. Each frame 1120 is described by two types of descriptors: a frame

descriptor 1140 giving information relating the packet; and, a pointer descriptor 1150 pointing to the buffer 1110 holding the data 1120. Each packet comprises a data payload 1170 preceded by a header 1160 in the same buffer 1180.

A write transaction 1190 to the Doorbell updates the number of words 1200 available for use by the adapter 80. This information is stored in the LCP context 140. The transmit LCP context 140 includes a pointer 1210 to the head of the buffer 1110 holding the data to be transmitted. When the LCP channel wins the internal channel arbitration of the ISOC 120, the ISOC 120 reads the descriptors of the LCP channel according to the pointer 1210 in the LCP context 140. Virtual addresses, for both descriptors 1100 and buffers 1110 of the LCP channel, are translated into physical addresses using the translation table 1130 located in the memory subsystem 210 of the adapter 80. The translation table 1130 is updated by the LCP manager 140 during registration of the memory buffers. The ISOC 120 reads the data and frame headers from the buffers 1110 into the adapter 80. The frame headers 1160 are then replaced on the ISOC 1320 by a header for the network 50. The packet header and the corresponding data are then transmitted to the network 50.

The RX LCP port is used to transfer incoming data from the ISOC 120 to the memory 60 used by a software application running on the host 10. TX LCP channels are completely controlled through descriptors initiated by the software on the host 10. RX LCP channels use descriptors from both the software on the host 10 and the ISOC 120. The descriptors initiated by the ISOC 120 are used to control the LCP channel operation to define the destination of a received frame in the memory 60 of the host 10. The descriptors initiated by the software in the host 10 can be used to define the location of buffers where the buffers were not defined through mapping in a translation table. To implement a handshake between the software in the host 10 and the adapter 80, two LCP channels are preferably used: an RX LCP channel for handling the received incoming data structure; and, an RX LCP channel for handling the completion status queue. The completion status is used by the adapter 80 to signal to the software in the host 10 that a frame transfer into the memory 60 of the host 10 is completed. Entries are inserted into the completion queue structure in sequential addresses. Each completion status entry contains a field that is marked by the adapter 80 and pooled by the software in the host 10 to check that the entry ownership has been transferred from the adapter

80 to the software in the host 10. One or more RX LCP channels can use the same completion status queue. The sharing of the completion status queue by multiple RX LCP channels is performed by the ISOC 120.

An RX LCP channel requires information to indicate the destination address for an incoming packet. The ISOC 120 has two addressing for finding the location of free buffers:

Direct addressing mode refers to LCP channels that do not use pointer descriptors to point out a buffer. The destination address is defined either by microcode in the ISOC 120 or read from the context 140.

Indirect addressing mode refers to LCP channels that maintain pointers to data buffers in descriptor structures. The descriptors are preferably located in the memory 60 of the host 10.

Direct addressing substantially cuts down the latency of processing an incoming packet through the adapter 80. However, it requires registration of memory buffer by the LCP manager 130, including storage of virtual to physical translation information on the adapter 80. The software in the host 10 writes to the channels Doorbell to indicate the amount of words added to the free buffer that can be used by the channel. In direct mode, the following steps are used to determine the address of the destination buffer:

A. Address A is driven as a command to the LCP engine.
B. (Optional) Address A is mapped to address A'.
C. Address A' (if step B is executed) or A (if step B is not executed) is the base address for the destination buffer.

In indirect mode, the adapter 80 uses descriptors to find the address of the data buffers. The descriptors are managed by the software in the host 10. The descriptors are preferably located in the memory 60 of the host 10. The term indirect is used to emphasize that the adapter 80 reads additional information to define the destination address. The adapter 80 accesses this information during run-time. Indirect addressing cuts down the amount of the memory in the

adapter 80 required to store translation tables. The descriptors are typically located in the memory 60 of the host 10. In indirect mode, the following steps are used to determine the address of the destination buffer:

A. Address A is driven as a command to the LCP engine.
B. (Optional) Address A is mapped to address A'.
C. Address A' (if step B is executed) or A (if step B is not executed) is the address of the pointer descriptor.
D. The pointer to the buffer, address B, is read from the descriptor.
E. (Optional) Address B is mapped to address B'.
F. Address B' (if step E is executed) or B (if step E is not executed) is the base address for the destination buffer.

Each RX LCP channel uses the following data structures:

Access to the Doorbell, implemented as a memory mapped address, informs the adapter 80 of additional data or descriptors available for the adapter 80 to write packet data.
An LCP context entry in the memory space 210 of the adapter 80 contains LCP attributes, state, configuration, and status fields.
Descriptors pointing to memory buffers for use in indirect mode.
A buffer in contiguous virtual address space in the memory 60 of the host 10.
A translation table and protection blocks in the memory space 210 of the adapter 80 for address mapping.

The flow of receiving a packet depends on the following characteristics:

Direct or indirect addressing mode.
For indirect mode, descriptors are located in the memory 60 of the host 10.
For direct mode, address mapping may or may not be used during access to descriptors.
Address mapping may or may not be used during access to buffers.

For indirect mode, address protection may or may not be used during access to descriptors.
Address protection may or may not be used during access to buffers.

These characteristics are set for each LCP channel as part of the channel's context 140 during the LCP channel initialization.

Referring to Figure 14, a flow of receive packets comprises, at step 1300, preparation by software 1310 in the host 10 of free buffer 1320 for the received data. At step 1330, in indirect mode, the software 1310 in the host 10 updates the descriptors 1340. The descriptors 1340 are located in the memory 60 of the host 10. At step 1350, the software in the host 10 rings the Doorbell to notify the adapter 80 of the free buffer space. For indirect mode, the Doorbell provides information indicative of the new descriptors 1340. For direct mode, the Doorbell provides information indicative of added free buffer space. At this stage, the adapter 80 is ready to transfer receive data from the network 30 to the memory 60 of the host 10. Steps 1300, 1330, and 1350 are repeated whenever the software 1310 in the host 10 adds free buffers 1320 to the RX LCP channel. The ISOC 120 repeats the following steps for each received packet. At step 1360, the adapter 80 receive the data. At step 1370, in indirect mode, the adapter 80 reads descriptors 1340 pointing to the location of the free data buffers 1320. At step 1380, data and headers are written into the data buffers 1340. At step 1390, status is updated in the completion queue.

Referring to Figure 15, in an example of a receive LCP channel flow, pointer descriptors are not used. Furthermore, no translation tables are used. Data buffers 1400 use contiguous space in the physical address space of software in the host 10 using the buffers 1400. Both header and data payload are written to the buffers 1400. A write transaction 1410 to the Doorbell updates the data space available for use by the adapter 80. The information is stored in the LCP context 140. The receive/completion LCP context 140 includes a pointer 1420 to the head of the buffer 1400 and an offset 1430 to the next/current address used to write new data/completion entries. When the adapter 980 receives a packet, it increments the offset 1430 to the next packet location and updates the available data space. A completion entry 1440 is added to a completion LCP 1450 upon completion of a frame reception, upon frame time-out,

or for any other frame event that requires awareness from the LCP client 100. The completion entry 1440 contains all the information needed by the LCP client 100 to locate the frame within the LCP data buffer 1400. The software in the host 10 uses a field within the completion entry 1440 to recognize that it has been granted ownership of the completion entry 1440.

The ISOC 120 allows LCP channels to be used for moving data between the memory subsystem 210 of the adapter 80 and the memory 60 of the host 10. To transfer data from the memory 60 of the host 10 to the adapter 80 a transmit channel is used. To transfer data from the adapter 80 to the memory 60 of the host 10 a receive channel is used. When data is to be transferred from the memory 60 of the host 10 to the adapter 80 a frame descriptor includes a destination address on the bus 340 of the ISOC 120. This address defines the destination of the frame data payload. The packet header is transferred in the usual manner. This allows loading of tables and code into the memory space of the ISOC 120. To transfer data from the memory space of the ISOC 120 to the memory 60 of the host 10 using a receive channel a descriptor is initiated by the RX processor 160. The descriptor include information indicative of both destination address in the memory 60 of the host 10 and source address.

In preferred embodiments of the present invention hereinbefore described, the adapter 80 is connected to the CPU 50 and memory 60 of the host computer system 10 via the bus architecture 70. However, in other embodiments of the present invention, the adapter 80 may be integrated into the host computer system 10 independently of the bus architecture 70. For example, in other embodiment of the present invention, the adapter 80 may be integrated into the host computer system via a memory controller connected to the host memory 60.

Additionally, in preferred embodiments of the present invention hereinbefore described, the adapter 80 was implemented in the form of a pluggable adapter card for insertion into the host computer system 10. It will however be appreciated that different implementation of the adapter 80 are possible in other embodiments of the present invention. For example, the adapter 80 may be located on a mother board of the host computer system, along with the CPU 50 and the memory 60.

CLAIMS

1. Apparatus for controlling flow of data between first and second data processing systems a memory, the apparatus comprising: descriptor logic for generating a plurality of descriptors including a frame descriptor defining a data packet to be communicated between a location in the memory and the second data processing system, and a pointer descriptor identifying the location in the memory; and a descriptor table for storing the descriptors generated by the descriptor logic for access by the first and second data processing systems.

2. Apparatus as claimed in claim 1, wherein the descriptor table is stored in the first data processing system.

3. Apparatus as claimed in claim 1, wherein the descriptor table is stored in the second data processing system.

4. Apparatus as claimed in any preceding claim, wherein the descriptor logic generates a branch descriptor comprising a link to another descriptor in the descriptor table.

5. Apparatus as claimed in claim 4, wherein the descriptor table comprises a plurality of descriptor lists sequentially linked together via branch descriptors therein.

6. Apparatus as claimed in claim 4, wherein the descriptor table comprises a cyclic descriptor list.

7. Apparatus as claimed in any preceding claim, wherein the first data processing system comprises a host computer system.

8. Apparatus as claimed in any preceding claim, wherein the second data processing system comprises a data communications interface for communicating data between the host computer system and a data communications network.

9. A data processing system comprising a host processing system having a memory, a data communications interface for communicating data between the host computer system and a data communications network, and apparatus as claimed in any preceding claim for controlling flow of data between the memory of the host computer system and the data communications interface

10. A method for controlling flow of data between first and second data processing systems via a memory, the method comprising: by descriptor logic, generating a plurality of descriptors including a frame descriptor defining a data packet to be communicated between a location in the memory and the second data processing system, and a pointer descriptor identifying the location in the memory; and storing the descriptors generated by the descriptor logic in a descriptor table for access by the first and second data processing systems.

11. A method as claimed in claim 10, comprising storing the descriptor table in the first data processing system.

12. A method as claimed in claim 10, comprising storing the descriptor table in the second data processing system.

13. A method as claimed in any of claims 10 to 12, comprising, by the descriptor logic, generating a branch descriptor comprising a link to another descriptor in the descriptor table.

14. A method as claimed in claim 13, comprising linking a plurality of descriptor lists together in series via branch descriptors to form the descriptor table.

15. A method as claimed in any of claims 10 to 14, wherein the first data processing system comprises a host computer system.

16. A method as claimed in any of claims 10 to 15, wherein the second data processing system comprises a data communications interface for communicating data between the host computer system and a data communications network.

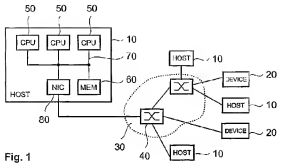

50
50
50
CPU
CPU
CPU
10
70
HOST
NIC
MEM
60
80
HOST
10
DEVICE
20
HOST
10
DEVICE
20
HOST
10
30
40

Fig. 1

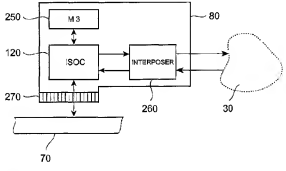

250
M3
80
120
SOC
INTERPOSER
270
260
30
70

Fig. 2

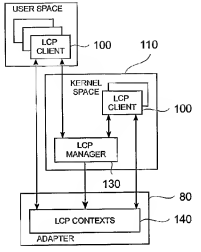

USER SPACE
LCP CLIENT
100
110
KERNEL SPACE
LCP CLIENT
100
LCP MANAGER
130
80
LCP CONTEXTS
140
ADAPTER

Fig. 3

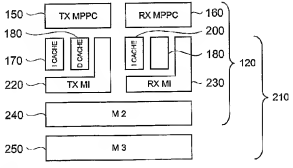

150
TX MPPC
RX MPPC
160
180
200
170
LOGIE
LOGME
LOACHE
180
120
220
TX MI
RX MI
230
210
240
M 2
250
M 3

Fig. 4

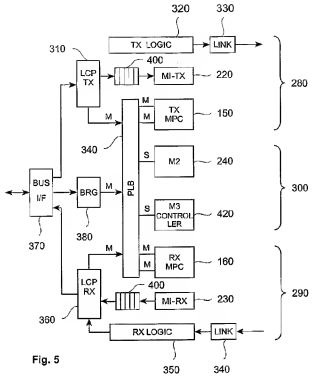

320
330
TX LOGIC
LINK
310
LCP TX
400
MI-TX
220
280
M
M
TX MPC
150
340
S
M2
240
BUS I/F
BRG
M
PLB
300
S
M3 CONTROL LER
420
370
380
M
M
RX MPC
160
LCP RX
400
MI-RX
230
290
360
RX LOGIC
LINK
350
340

Fig. 5

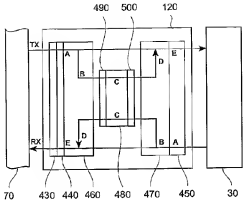

490
500
120
TX
A
B
C
C
D
D
E
E
RX
B
A
70
430
440
460
480
470
450
30

Fig. 6

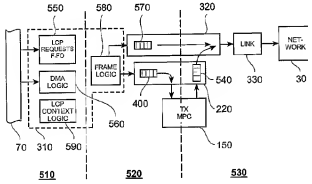

550
580
570
320
LCP REQUEUE'S FIFO
DMA LOGIC
LCP CONTEXT LOGIC
FRAME LOGIC
LINK
NET-WORK
540
330
30
400
220
TX MPC
560
150
70
310
590
510
520
530

Fig. 7

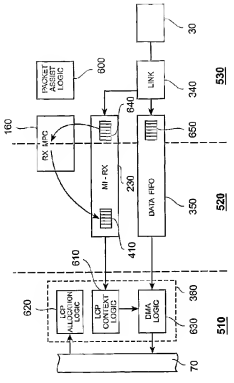

30
PACKET FILTER LOGIC
600
LINK
340
530
640
160
RX MPC
650
MI-RX
230
DATA FIFO
350
520
410
610
620
LCP LOCATION LOGIC
LCP CONTEXT LOGIC
DMA LOGIC
380
630
510
70

Fig. 8

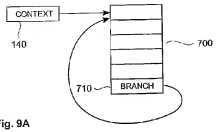

CONTEXT
140
700
710
BRANCH

Fig. 9A

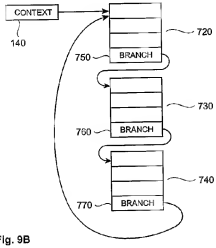

CONTEXT
140
720
750
BRANCH
730
760
BRANCH
740
770
BRANCH

Fig. 9B

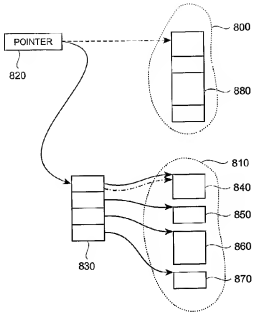


Fig. 10

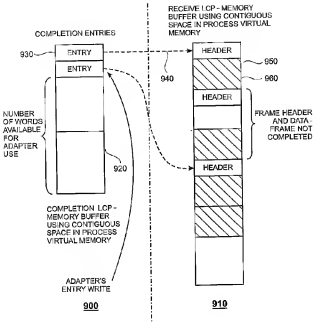

RECEIVE LCP - MEMORY BUFFER USING CONTIGUOUS SPACE IN PROCESS VIRTUAL MEMORY
COMPLETION ENTRIES
930
ENTRY
ENTRY
940
HEADER
950
960
HEADER
FRAME HEADER AND DATA FRAME NOT COMPLETED
NUMBER OF WORDS AVAILABLE FOR ADAPTER USE
920
HEADER
COMPLETION LCP - MEMORY BUFFER USING CONTIGUOUS SPACE IN PROCESS VIRTUAL MEMORY
ADAPTER'S ENTRY WRITE
900
910

Fig. 11

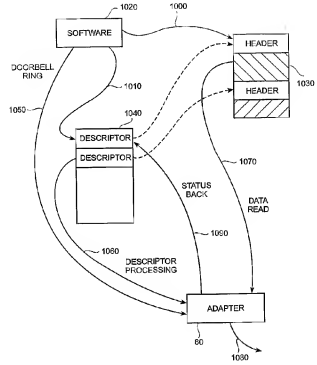

1020
1000
SOFTWARE
HEADER
DOORBELL RING
1010
1030
HEADER
1050
1040
DESCRIPTOR
DESCRIPTOR
1070
STATUS BACK
DATA READ
1060
1060
DESCRIPTOR PROCESSING
ADAPTER
60
1030

Fig. 12

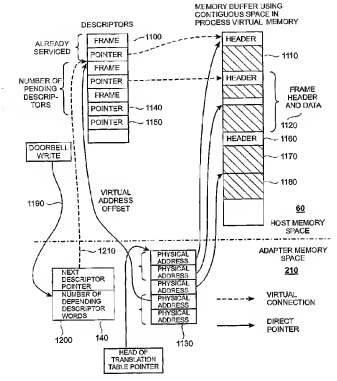

MEMORY BUFFER USING CONTIGUOUS SPACE IN PROCESS VIRTUAL MEMORY
DESCRIPTORS
ALREADY SERVICED
NUMBER OF PENDING DESCRIP-TORS
FRAME
POINTER
FRAME
POINTER
FRAME
POINTER
POINTER
1100
1140
1150
HEADER
HEADER
HEADER
1110
FRAME HEADER AND DATA
1120
1160
1170
1180
DOORBELL WRITE
1190
VIRTUAL ADDRESS OFFSET
60
HOST MEMORY SPACE
1210
PHYSICAL ADDRESS
PHYSICAL ADDRESS
PHYSICAL ADDRESS
PHYSICAL ADDRESS
PHYSICAL ADDRESS
PHYSICAL ADDRESS
ADAPTER MEMORY SPACE
210
NEXT DESCRIPTOR POINTER
NUMBER OF DESCRIPTOR WORDS
VIRTUAL CONNECTION
DIRECT POINTER
1200
140
1130
HEAD OF TRANSLATION TABLE POINTER

Fig. 13

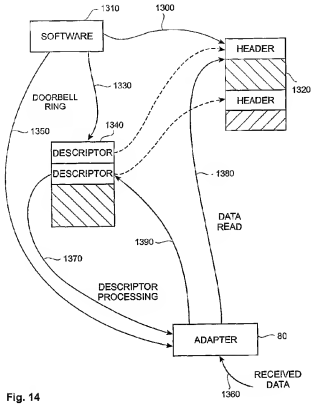

1310
1300
SOFTWARE
HEADER
1320
HEADER
1330
DOORBELL RING
1350
1340
DESCRIPTOR
DESCRIPTOR
1380
DATA READ
1390
1370
DESCRIPTOR PROCESSING
ADAPTER
80
RECEIVED DATA
1360

Fig. 14

WO 02/061592 PCT/IB01/00122

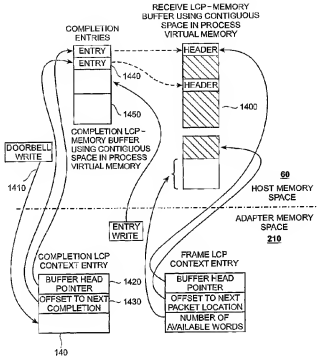


Fig. 15

【国際調査報告】

INTERNATIONAL SEARCH REPORT

Int ernational Application No
PCT/US01/00122

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F13/40

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 359 730 A (MORROW ASSAF) 25 October 1994 (1994-10-25) column 9, line 3 -column 4, line 18 column 8, line 19 -column 10, line 23 abstract	1-16
A	US 6 028 843 A (LEICHTY PHILIP LYNN ET AL) 22 February 2000 (2000-02-22) column 1, line 26 -column 2, line 17	1-16
A	EP 0 470 876 A (THOMSON CSF) 12 February 1992 (1992-02-12) the whole document	1-16
A	US 5 933 632 A (CANTILL III BENJAMIN M) 3 August 1999 (1999-08-03) column 2, line 20 - line 40 abstract	1-16

Further documents are listed in the continuation of box C.

X Patent family members are listed in annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search
30 November 2001

Date of mailing of the international search report
06/12/2001

Name and mailing address of the ISA
European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,
Fax: (+31-70) 340-3016

Authorized officer
Nguyen Xuan Hiep, C

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International Application No
PCT/GB 01/00122

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5359730	A 25-10-1994	NONE	
US 6028843	A 22-02-2000	GB 2324678 A ,B	28-10-1998
		GB 2341751 A ,B	22-03-2000
		JP 3065983 B2	17-07-2000
		JP 10276221 A	13-10-1998
EP 0470876	A 12-02-1992	FR 2665040 A1	24-01-1992
		DE 69107727 D1	06-04-1995
		DE 69107727 T2	13-07-1995
		EP 0470876 A1	12-02-1992
		ES 2069241 T3	01-05-1995
US 5553632	A 03-09-1996	NONE	

Annex PCT/ISA/210 (patent family annex) (July 1992)

フロントページの続き

(72)発明者　ビラン、ギオラ
イスラエル　ズイクラン・ヤアコヴ　インバル・ストリート　１３
(72)発明者　ソストハイム、タル
イスラエル　キルヤト・ティヴォン　３６０００　アロニム・ストリート　７

Ｆターム(参考) 5B089 GA04 KA05 KC15 KE09
5K030 GA01 HA08 HB13 LC01